VŠB-Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2012/2013 Bc. Jiří Haška

VŠB-Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a biomedicínského inženýrství

Zpracování, analýza a detekce obrazového signálu na robotickém zařízení
Image Signal Processing, Analysis and Detection for Robotic System

2012/2013 Bc. Jiří Haška

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Jiří Haška**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Zpracování, analýza a detekce obrazového signálu na robotickém zařízení
Image Signal Processing, Analysis and Detection for Robotic System

Zásady pro vypracování:

1. Rozbor problematiky algoritmů a metod pro zpracování, analýzu a detekci obrazového signálu.
2. Návrh a realizace metod pro analýzu obrazového signálu s detekcí barev, tvarů a lidských obličejů.
3. Řešení detekčního a řídicího systému robotického zařízení.
4. Praktické ověření a testování robotického zařízení a jednotlivých implementovaných metod.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] MANN, Burkhard. *C pro mikrokontroléry : uC & praxe.* Překlad Václav LOSÍK. 1. vyd. Praha : BEN - technická literatura, 2003. 280 s., 1 CD-ROM. ISBN 80-7300-077-6.
[2] HRBÁČEK, Jiří. *Kommikace mikrokontrolérů : s okolím 2.* 1.vyd. Praha : BEN-technická literatura, 2000. 151 s. ISBN 80-86056-73-2.
[3] MAREŠ, Amadeo. *1001 Tipů a triků pro C#.* 1. vyd. Praha : Computer Press, a.s., 2008. 360 s. ISBN 978-80-251-2125-2.
[4] ROZEHNAL, Zdeněk. *Mikrokontroléry MOTOROLA : HC11.* 1.vyd. Praha : BEN-technická literatura, 2001. 191 s. ISBN 80-86056-77-5.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Zdeněk Macháček, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 19.07.2013

doc. Ing. Jiří Kozlorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Haška

.....

Bc. Jiří Haška

Datum odevzdání diplomové práce: 19. 7. 2013

Poděkování

Touto cestou bych chtěl poděkovat vedoucímu mojí diplomové práce panu Ing. Zdeňkovi Macháčkovi, Ph.D. za poskytnutí materiálů, cenných rad a konzultací, které mi ochotně během tvorby diplomové práce poskytly.

Abstrakt

V našem světě se objevuje stále víc a víc robotických zařízení. Některá z těchto zařízení potřebují sledovat okolní svět. Na to je nejvhodnější používat kamery. Pomocí složitých algoritmů se dá z načtených dat vyhledat a zaměřit téměř vše. Od detekce barev, tvarů až po detekci a rozpoznání lidí.

Klíčová slova

Robot, robotické zařízení, robotická hlava, kamery, kamerové senzory, senzor, detekce tvarů, detekce barev, detekce lidských obličejů.

Abstract

In our world appear more and more robotic devices. Some of these facilities need to monitor the outside world. It is best to use the camera. Using complex algorithms can be retrieved from the data search and focus almost everything. Since the detection of colors, shapes to detect and identify people.

Keywords

Robot, a robotic device, a robotic head, camera, camera senzors, senzors, face detection, color detection, detection of human faces.

Seznam použitých zkratek a symbolů

CAN	- (Controller-area network)Zkratka pro komunikační protokol
CRC	- (Cyklic Redundand Control) Cyklická redundantní kontrola
IIC	- (Inter-Integrated Circuit) Zkratka pro komunikační protokol I2C
RGPIO	- Rychlý univerzální vstup / výstup
NTSC	- (National Television System Committee) Analogový obrazový formát
PAL	- (Phase Alternationby Line) Analogový obrazový formát
MCG	- Univerzální generátor hodin
CIF	- (Common Intermediate Format) Digitální obrazový formát
VGA	- (Video Graphics Array) Nejpoužívanější digitální obrazový formát
SIF	- (Standard Interchange Format) Digitální obrazový formát
KBI	- Klávesnice přerušení
CCD	- (Charge Coupled Device) Technologie pro obrazový senzor
CMOS	-(Complementary metal-oxide-semiconductor)Technologie pro obrazový senzor
CIE	- (International Commission on Illumination) První matematický definovaný barevný prostor

Seznam použitých obrázků

Obsah

1. Úvod

Vzhledem ke stále se zdokonalujícím robotickým zařízením, jejich složitosti a možnosti pohybovat se, rozvíjí se jejich potřeba sledovat okolní svět. Je nutné rozhodnout, jakým směrem se vy vývoji jejich očí vydáme. Jelikož jsme náš svět vytvořili k našemu obrazu, bylo by vhodné, aby i roboti viděli podobně jako lidé. K tomu je nejlepší používat kamery.

Takovéto výhody již dnes nedisponují pouze roboti sledující své okolí, ale i inteligentní automobily sledující značky, ospalost řidiče a jiné bezpečnostní hrozby. Ve městech se dá tato technologie využít pro vyhledání osob anebo předmětů.

S tímto je spjat problém výkonu. Zkvalitnění obrazu přináší snadnější rozpoznávání a detekci, ale tím prohledání obrazu zpomalí. Je tedy nutné přizpůsobit této výpočtové náročnosti i hardware.

V kapitole Rozbor problematiky algoritmů a metod pro zpracování, analýzu a detekci obrazového signálu se zabývám teoretickými problémy kamerových senzorů. Jejich porovnávání s ostatními typy na trhu, jeho osazení optikou a typem optiky. Je zde porovnáván i použitý mikroprocesor, jeho alternativy a možnosti využití a synchronizace a čtecí sekvence z kamerových senzorů.

Další kapitolou je návrh systému. Jelikož je celá robotická hlava prací čtyř jednotlivých částí, je v této kapitole pojednáváno, jak na sebe budou práce navazovat. Návrh pohybových možností, komunikačních možností a k nim adekvátní odpovědi ze strany kamerových senzorů. Dále je v této kapitole návrh graficko-uživatelského rozhraní.

V kapitole Realizace systému jsou tyto návrhy detailně popsány a vysvětleny. Na začátku kapitoly je vysvětleno graficko-uživatelského rozhraní. Jeho princip a jeho tvorba ve Visual Studiu. Dále, jsou zde popsány možnosti odesílání zpracovaných dat a nastavení rozhraní při přijetí obrazu. A vytvořený hardware, jak pro desku s kamerovým senzorem, která musí být vytvořená co nejmenší, aby se dala vložit do robotické hlavy a pohybovat s nimi, tak i deska s procesorem pro univerzální kit.

Další kapitola, nazvaná Metody pro analýzu obrazového signálu s detekcí barev, tvarů a lidských obličejů, pojednává o algoritmech, vyvinutých pro detekci snímaných obrazů. Od barev, přes detekci středu, počtu objektů, detekci tvarů, až po detekci lidského obličeje. V této kapitole je nejvíce popsán algoritmus v jazyce C, ke kterému je vytvořen vždy UML diagram. A popis funkcí jednotlivých částí.

V poslední kapitole, Měření a verifikace je popsáno testování celé aplikace. Od univerzální deky, její rychlosti zpracovávající přijímaných dat, přes rychlost kamerového senzoru až po měření a testování sejmutého obrazu v PC. Bude zde otestován algoritmus detekce barev a vyhledávání objektů v obraze a přesnost detekce lidských obličejů v různých prostředích a osvětleních.

Celá diplomová práce se bude ubírat směrem takovým, aby finální výsledný prototyp robotické hlavy komunikoval s okolním prostředím podobně jako by to byl člověk. Tedy v případě, že si před ní objeví člověk, hlava se na něj podívá a bude na něj reagovat.

2. Rozbor problematiky algoritmů a metod pro zpracování, analýzu a detekci obrazového signálu.

V této kapitole jsou porovnávány použité kamerové senzory s ostatními na trhu, jejich osazení optikou a výběr optiky. Nastavení kvality obrazu a porovnávání použitého mikrokontoléru s ostatními. Jeho zapojení na desku a jeho maximální možnosti v oblasti výpočetní rychlosti algoritmů.

Celý koncept robotického zařízení se bude chovat tak, že uživatel řekne například větu: Spočítej zelené tvary. Tuto informaci převezme snímání zvuku. Přeloží ji, zpracuje a odešle po sběrnici informaci o tom, co je po robotu požadováno. Rozhodovací protokol rozhodne, že tato informace je určena pro kamery a odešle ji na kontrolér zpracovávající obraz. Ten obraz rozloží na barvy a vybere pouze zelené, ty změní v bílou a ostatní v černou. Spočítá a tuto informaci odešle po sběrnici jako odpověď. A robot odpoví například čtyři.

2.1. Vnímání obrazu v různých vlnových délkách

Obraz, který je pořízen jako digitální fotografie, může být v různých oblastech spektra. Může být ve viditelném spektru, jako třeba denní světlo, anebo v oblastech lidskému oku neviditelné. Jako třeba infračervené snímky, rentgenové snímky apod. Tyto rozdíly je třeba brát v úvahu, protože chceme docílit, aby se na snímek vlezlo co největší množství informací.

Světlo je elektromagnetické vlnění, které je v podstatě dvojdimenzionálním signálem. Prostupnost prostřední je také důležité k získání kvalitního obrazu. Snímek v infračerveném spektru v mlze, zachytí něco jiného ve viditelném spektru. Zato radarové snímky, používající radarové vlny, zachytí i pohoří přes husté mraky. A trochu jiné jsou i ultrazvukové snímky. [8]

Platí, že frekvence, rychlost šíření vlnění a vlnová délka zařízení jsou spolu ve vzájemném vztahu:

$$f = c/\lambda \qquad \text{Rovnice 1}$$

Mezi frekvencí elektromagnetického vlnění fa energií fotonu E při dané vlnové délce platí vztah:

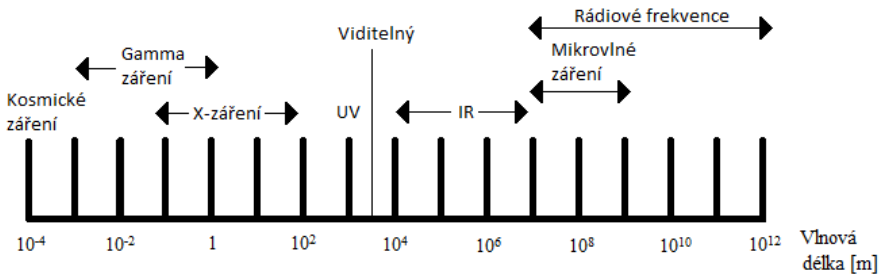
$$E = hf \qquad \text{Rovnice 2}$$

Kde: h = 6,625 * 10-34 [J*s] … Planckova konstanta

Převodní vztah pro energii fotonu:

$$E[eV] = E[J]/e \qquad \text{Rovnice 3}$$

Kde: e = 1,602 *10-19 … náboj elektronu



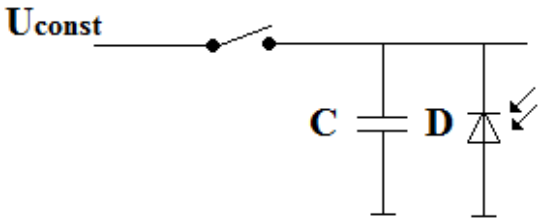
Obrázek 1. Elektromagnetické spektrum[9]

2.2. Digitalizace obrazu

V reálném světě, lze obraz chápat jako spojitý, tedy analogový signál. V technickém světě se tyto informace ukládají jako diskrétní. Proto po převedení spojitého signálu na diskrétní vznikají jisté rozdíly. Nejčastějším převodem jsou digitální fotografie nebo scanování. Takovýmto způsobem se převede každý kousek obrazu na jednotlivé body nazývané pixlely. Každý pixel nese informace o bodu, jako je barva a jas.

Nejčastějším způsobem jak převést obraz na digitální je vystavit ho senzoru. Obrazové senzory se v základu dělí na CMOS a CCD jejich vlastnosti mohou velmi ovlivnit výsledek snímání.

Princip činnosti je založen na převodu obrazu promítaného na plochu senzoru. U CCD senzorů se elektrický náboj přesouvá CCD strukturami na okraj senzoru a následně je převeden na napětí. U CMOS senzorů se hodnota náboje převání na napětí přímo v jednotlivých buňkách. [8]



Obrázek 2. Ideové schéma světlocitlivé struktury

2.3. Formáty pro ukládání obrazu

Snímaný obraz, který chceme ukládat do paměti, pevný disk nebo jiné médium, může být vektorový nebo rastrový. Digitální fotoaparáty používají rastrové formáty. Tyto formáty mají vysoké nároky na paměť.

K uložení nekomprimovaného obrazu z pěti megapixelového fotoaparátu musíme mít velikost paměti:

Velikost obrazu:2592 x 1944 bodů

1Byte = 8bitů (pro reprezentaci RGB)

Pokud je každý pixel ve 24bit barevné hloubce tak na každý pixel jsou 3byte paměti.

3 x 5 = 15 MB Rovnice 4

Je tedy třeba mít k dispozici 15MB paměti. Proto se ke snížení nároků na paměť využívá kompresí. Mohou být dvojího typu, ztrátová a bezeztrátová.

Bezeztrátové formáty: TIFF, RAW, PNG, MNG.

Ztrátové komprese jsou založeny na nedokonalosti lidského oka, ale po zrekonstruování obrazu zpět, nezískáme originál nýbrž ořezanou verzi. [8]

2.4. Barevné modely

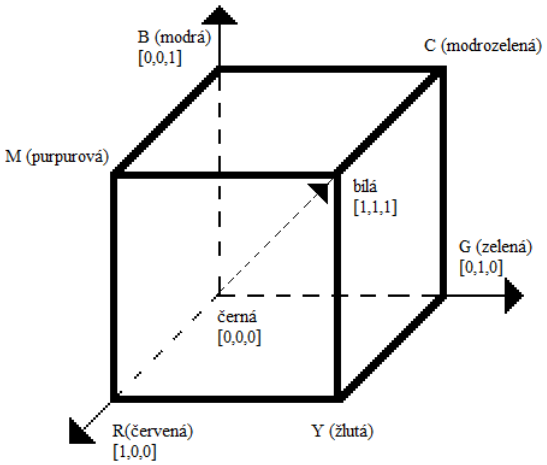
Lidské oko dokáže vnímat rozsah spektra přibližně od 400 do 700nm. Pokusy s rozkladem slunečního světla prováděl už Isaac Newton v roce 1666 za pomoci skleněného hranolu. Avšak základy kolorimetrie (barevné teorie) byly položeny v roce 1931 CIE.

Mezi nejvýznamnější barevné modely patří bezesporu RGB (Red, Green, Blue -červená, zelená, modrá) a CMY (Cyan, Magenta, Yellow - azurová, purpurová, žlutá). Nevýhodou těchto barevných modelů je závislost na zařízení a komplikovanější nastavení odstínů barev. Pro lidské oko je vhodnější model, kde je oddělená jasová (luminiscenční) a barvonosná (chrominační) složka.

2.4.1. Barevný model RGB

Tento model je nejčastějším modelem používaný v zobrazovacích a výstupních zařízeních jako je televize a monitor.

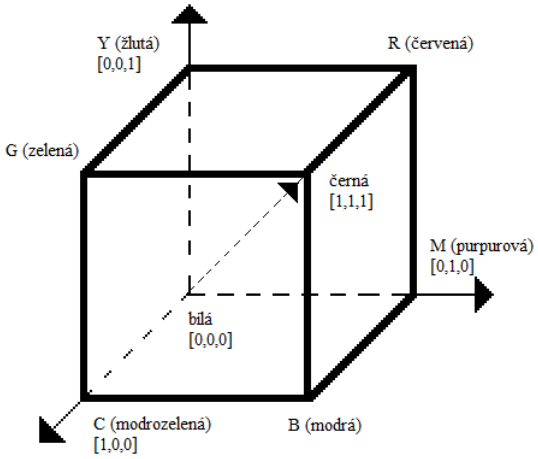
Jednotlivým složkám byly přiřazeny vlnové délky: red = 700nm, green = 546,1nm a blue = 435,8nm. Každá barva je reprezentována jistým počtem bitu. Většinou to bývá 8bitů. V patřičném poměru těchto složek dosáhneme požadované barvy. Při 3*8bitech lze vyjádřit 16,7 milionů barev. (23*8 = 224). Barevný prostor RGB se označuje jako aditivní skládání barev (čím vyšší hodnoty tím světlejší). Černá barva je v krychli reprezentována jako [0, 0, 0] což je vrchol a všechny odstíny jsou na minimu.



Obrázek 3. Reprezentace barevného prostoru RGB pomocí krychle

2.4.2. **Barevný model CMY**

Tento model se také nazývá subtraktivní. To je stejná obdoba jako při překrývání barev u malířských technik. Tento model se používá především u tiskáren pro tisk. U tohoto modelu je černá barva reprezentována jako [1, 1, 1] což je opět vrchol, ale všechny odstíny jsou na maximu.



Obrázek 4. Reprezentace barevného prostoru CMY pomocí krychle

2.5. Princip prohledání obrazu

Pokud máme obraz snímaný v reálném čase, je velmi obtížné a na výpočetní výkon velmi náročné detekovat různé objekty. Proto je dobré vytvořit stacionární obraz a z něj detekovat objekty a provádět nejrůznější výpočty.

Obraz lze například uložit do matice a v ní provádět hledání. Aby bylo možné, matici vůbec vytvořit je nutné použít dva do sebe vnořené Fory. [10] A to následujícím způsobem.

```
for (int i = 0; i < zdroj.výška; i++)          //For pro výšku
{
      for (int j = 0; j < zdroj.šířka; j++)    //For pro šířku
     {

     }
}
```

Při prohledávání matice nebo obrazu se nejprve přičte proměnná i a tím se prohledá první pixel. Kód nyní vleze do druhého foru, kde bude prohledávat každý pixel na řádku, dokud nedojede na konec řádku. Poté vyleze z vnořeného foru zpět do prvního, kde se přičte o jedna proměnná a přejde se v matici nebo v obraze na druhý řádek. Takovýmto způsobem se prohledává, dokud není poslední řádek.

2.6. Kamerový senzor MT9V131

Kamerový senzor je bezpochyby základem celé problematiky analýzy obrazu. Pro toto robotické zařízení byl vybrán senzor s označením MT9V131.

Je to levný, dostupný a jeho parametry nejsou natolik náročné pro použitý mikročip. Je to klasické VGA s rozlišením 640H x 480V. Velikost jednoho pixelu je 5,6µ x 5,6µ. Senzor je RGB, s napájecím napětím 2,8V s výkonem 80mW. Maximální přenosová rychlost je 12-13,5 Mp/s. U tohoto senzorů lze snížit FrameRate z VGA na CIF (352 x 240) nebo na QVGA (320 x 240).

2.7. Výběr rozlišení

Rozdíl mezi analogovým a digitálním obrazem je v definování rozlišení. Zatímco u analogového obrazu se počítají řádky, u digitálního se počítají pixely.

2.7.1. Typy rozlišení

Pro analogová videa se používá NTSC a PAL. Oba tyto standardy se používají především v televizním průmyslu.

NTSC – 480 řádků a 30fps

PAL – 576 řádků a 25fps

Celkový přenos dat je však stejný. Po digitalizaci videa dostáváme NTSC 704x480 a PAL 704x 576

V zabezpečovacím průmyslu, se stal znám standard CIF, který má velikost ve formátu NTSC 352×240 pixelů, ve formátu PAL 352×288 pixelů.

Pro digitální videa, představována především digitálními fotoaparáty a kamerami jsou analogová videa bezpředmětná. Pro počítačový průmysl vzniky nové celosvětové standardy.

Nejznámějším a snad nejpoužívanějším je VGA standard. Ten má velikost podobnou NTSC a PAL, ale je daleko vhodnější pro digitální snímače protože jejich formát je v mnoha případech podobný počítačovým monitorům. Někdy se v digitálních videích používají jeho poddruhy a násobky jako např. QVGA a CIF. QVGA se někdy označuje jako SIF což se plete s CIF.[3]

Rozlišení: VGA 640 x 480

QVGA 320 x 240

XVGA 1024 x 768

CIF 352 x 240

Dalším formátem je 4 násobek VGA známý jako MPEG.

Rozlišení: TV NTSC 704×480

TV PAL 704×576

DVD-Video NTSC 720×480
DVD-Video PAL 720×576

2.7.2. **Porovnání rozlišení**

Kamerový senzor podporuje pouze tři digitální rozlišovací módy a to VGA, QVGA a CIF. Jejich rozlišení a tedy kvalita je zřejmá z obrázků. Z obrázku je zřejmé, že VGA, které má rozlišení 640x480 je nejkvalitnější. Proto je i v této práci preferován a používán. Ostatní dva formáty nejsou pro detekci v obraze vhodné. Zvlášť na větší vzdálenosti.

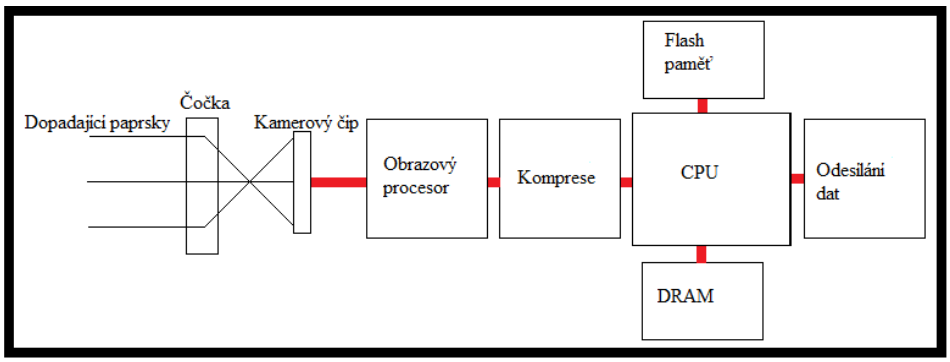
Obrázek 5. Porovnání kvality (zleva VGA, CIF, QVGA)

2.8. Optika

Kamerový senzor je nutné osadit optikou. Proto je nutné s tímto počítat a desku plošných spojů navrhnout tak, aby ho tam bylo možné připevnit. Po dokončení a načtení prvních pixelů, je nutné nastavit optiku tak, aby obraz nebyl rozmazaný. Proto bude nutné toto ještě testovat a vybrat správné rozlišení.

2.8.1. Použití optiky

Při snímání obrazu se před kamerovým senzorem používá optika. Úkolem optiky je, aby příchozí fotony zachytil a správně odeslal na snímací senzor. Snímací senzor je převede na elektrický signál, který je dále zpracováván například kontrolérem. [4] Jak je vidět na následujícím obrázku.



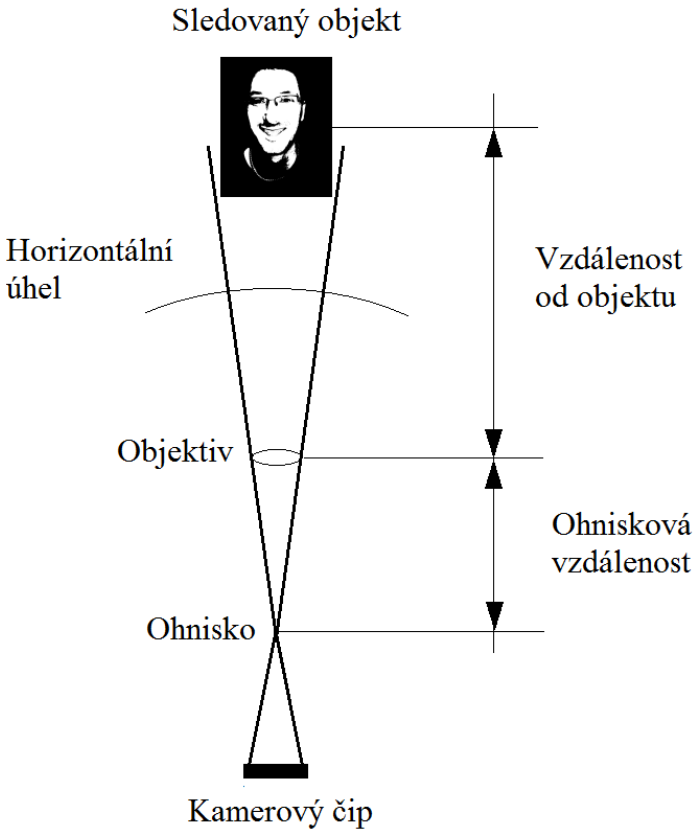
Obrázek 6. Průřez IP kamerou

2.8.2. Ohnisková vzdálenost

Ohnisková vzdálenost je úhel záběru neboli zorné pole optiky. Čím více roste ohnisková vzdálenost, tím je pozorovací úhel nižší (užší) a tím více se nám snímaný objekt zdánlivě přibližuje a vidíme tedy více detailů[4] Tato činnost se nazývá analogové zoomování. Dále ještě existuje digitální zoomování, ale to se provádí v senzoru.

2.8.3. **Rozlišení a úhel záběru**

Je nutné při výběru vycházet z účelu pořízení obrazu. Zdali se jedná o identifikaci obličeje anebo zachycení pohybu před kamerou. V této práci se jedná o identifikaci předmětů, barev a lidských obličejů. Po tomto rozhodnutí je tedy nutné dohledat ohniskové vzdálenosti a úhel záběru. K tomuto účelu je možné použít tabulku[7]. Anebo ohniskovou vzdálenost a úhel záběru dopočítat následujícím způsobem[6].



Obrázek 7. Úhel záběru [5]

Pro vzorce platí:

Šířka objektu – H
Vzdálenost od objektu – D
Ohnisková vzdálenost – f
Horizontální konstanta – h
Vertikální konstanta – v

Řekneme, že robotické zařízení bude od snímaného objektu vzdáleno 3 metry a šíře snímaného objektu bude 2metry.

Pak ohnisková vzdálenost bude:

$$f = h \times D/H \qquad \text{Rovnice 5}$$

A úhel záběru:

$$\alpha = 2tan^{-1}\left(\frac{h}{2f}\right) \qquad \text{Rovnice 6}$$

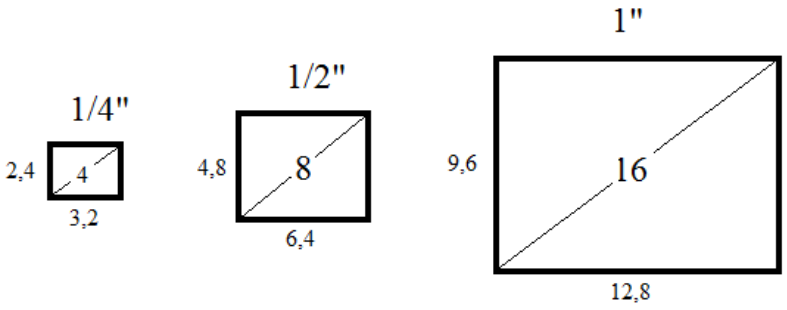
Pro upřesnění jsou v následující tabulce doplňující informace k jiným vybraným objektivům a vertikální konstanta. Tyto hodnoty se dají použít se stejnými vzorci.

Tabulka 1. Vertikální a horizontální konstanty

Formát objektivu	1/4"	1/3"	1"
v	2,7	3,6	2,7
h	3,6	4,8	3,6

2.8.4. **Velikost optiky**

Kamerový senzor MT9V131 má optický formát 1/4" to je velikost 4:3.



Obrázek 8. Porovnání velikosti optiky

Při výběru optiky je důležité mít na paměti, že je nutnost mít optiku stejnou anebo větší než je formát kamerového senzoru. Pokud by se vybrala optika menší, než formát výsledný obraz byl orámovaný, protože optika nepokrývá celou plochu snímače. Ale naopak, kdyby byla

optika větší než formát senzoru, byl by obraz kvalitnější, ale zároveň by byl úhel záběru menší než při použití stejné optiky.[6]

2.9. Výběr kamerového senzoru

Poměr cena/výkon v porovnání s podobnými typy jako námi vybraný typ MT9V131.

S těmito stejnými parametry:

Res.	VGA
OpticalFormat	1/4inch
FrameRate	30 fps
Pixel Size	5.6µm
Supply Voltage	2.8V
Master Clock	27 MHz

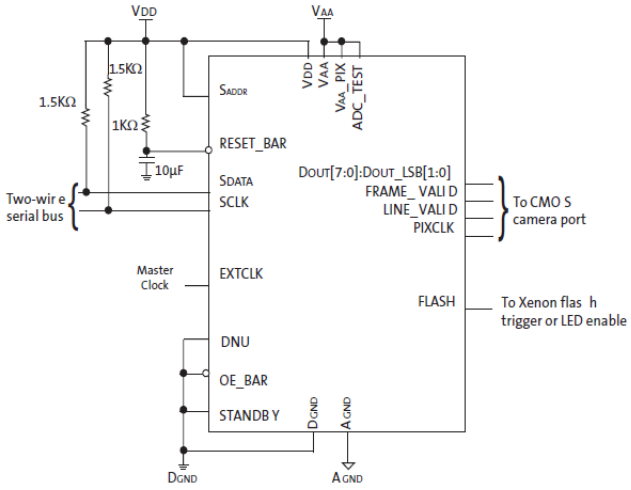
Tabulka 2. Porovnání kontrolérů

Název	Cena [Kč]	Imaging Area [mm]	DynamicRange [dB]	Responsivity [V/lux-sec]	Data Rate [Mp/s]	Data Format	Power [mW]
MT9V125IA7XTC	753,30	3.63x 2.78	70	5	13.5	10-bitparallel, NTSC/PAL composite video	320
MT9V131C12STC	601,82	3.58 x 2.69	60	1.9	13.5	10-bitparallel	<80
MT9V136C12STC	538,70	3.58 x 2.69	82	11.5	27	8-10-bit paralleldigital output	300
MT9V135C12STC	486,71	3.63 x 2.78	70	5	13.5	8-bit/10-bit-serial and parallel, NTSC/PAL	320

Byl vybrán typ MT9V131. V porovnání s ostatními podobnými typy má menší dynamický rozsah, horší elektrický výkon za optickým vstupem a srovnatelnou rychlost přenosu dat. Má menší spotřebu elektrické energie, což se ocení zejména v bateriovém módu. V tabulce není porovnávána dostupnost, ale tento typ je poměrně dobře dostupný. Pro účely tohoto robotického zařízení je senzor dostačující, i když při porovnání ceny s výkonem by mohl být levnější. Celá tabulka porovnání cena / výkon a s typy dostupnými na dnešním trhu je v přílohách.

2.10.Zapojení kamerového senzoru

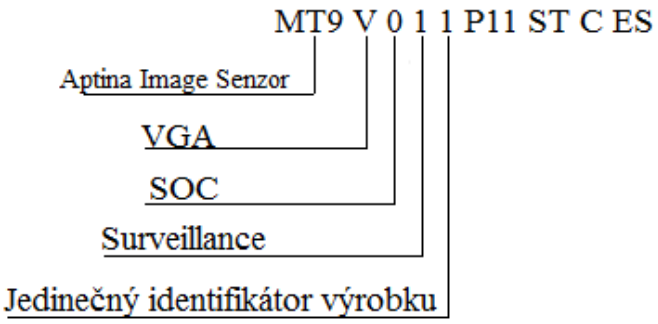
Zapojení kamerového senzoru je na obrázku č. 9. Ten byl doporučený výrobcem jako nejvhodnější. Avšak pro tento robotický systém, bylo schéma upraveno a některé součástky byly vypuštěny a cesty upraveny. Bylo to z důvodu obav ze statické elektřiny a ztrát napětí na delších cestách.



Obrázek 9. Doporučené zapojení kamerového senzoru

2.11.Označení kamerového senzoru

Kamerový senzor je do firmy Aptina. Tento výrobce vytváří více druhů kamerových senzorů. Na obrázku, je vidět dělení a možnosti kamerových senzorů z rodiny MT9V.Celý obrázek je v příloze.



Obrázek 10. Dělení kamerových senzorů

2.12.Čtecí sekvence a synchronizace pro kamerový senzor MT9V131

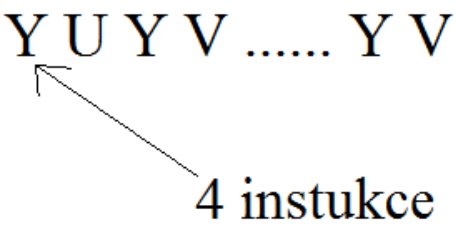
Rychlost procesoru je 32MHz, avšak rychlost nutná pro snímání obrazu je 12MHz. Toto je náno 16MHz krystalem připojeným k mikroprocesoru a jeho dělička dává na výstup těch potřebných 12MHz.

Je potřeba 4 instrukce procesoru na data kamery

V případě, že je potřeba načítat černobílý obraz, tak stačí načítat každou čtvrtou instrukci, přicházející z kamerového senzoru.

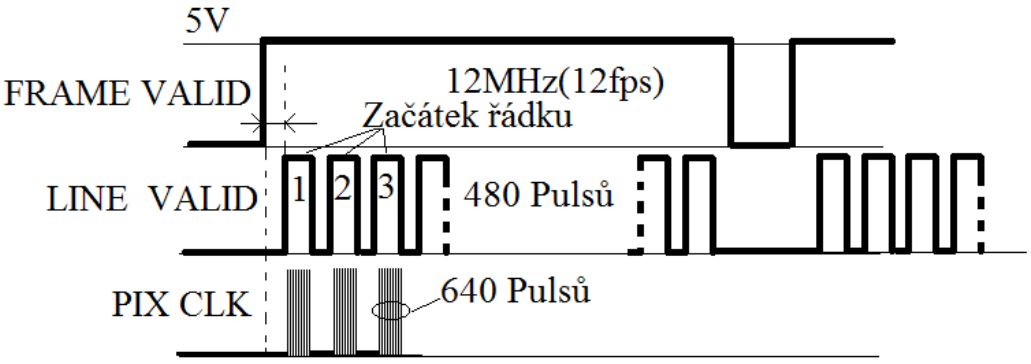
Y U Y V Y U Y V Y U Y V Y U Y V Y U Y V Y U Y V Y U Y V Y U Y V

Takto vypadá datový výstup z kamerového senzoru.



Obrázek 11. Popis instrukcí z výstupu kamerového senzoru

Na kamerový senzor je přiveden CLK. Ten kamerový senzor zpracuje. Signál přicházejícího z kamerového senzoru je označován jako FRAME_VALID a má frekvenci 15Hz. A označuje začátek snímku. Na náběžnou hranu tohoto signálu reaguje pin označovaný jako LINE_VALID. Ten má frekvenci 7KHz a obsahuje 480 pulsů tvořící informaci o řádcích. Když je LINE_VALID v log 1 je na výstupu z kamerového senzoru označovaném jako PIX 640 pulzů nesoucí informaci o jednotlivých pixelech. [1]



Obrázek 12. Odesílání dat z kamerového senzoru

2.13.Přepočet ze složek YUV na RGB

Na výstupu kamerového senzoru je barevný model YUV. Kde Y je jasová složka a U a V jsou barevné složky. Pro detekci barev, tvarů a lidských obličejů je vhodnější mít RGB model. Pro přeměnu YUV barevného modelu na RGB barevného modelu se používají následujících rovnic [12].

$$R = 1{,}164(Y - 16) + 1{,}596(V - 128) \quad \text{Rovnice 7}$$

$$G = 1{,}164(Y - 16) - 0{,}813(V - 128) - 0{,}391(U - 128) \quad \text{Rovnice 8}$$

$$B = 1{,}164\,(Y - 16) + 2{,}018\,(U - 128) \quad \text{Rovnice 9}$$

2.14.MicrocontrollerMCF51AC256

Tento mikrokontrolér od firmy Freescale Semiconductor s označením MCF51AC256 je založen na ColdFire jádře a toto jádro funguje při rychlostech až do 50, 33 MHz. Má 256KB flash paměti, 32 statické paměti RAM (SDRAM), dva analogové komparátory (ACMP), analogově-digitální převodník až na 24 kanálu, sběrnici CAN, CRC, IIC, KBI, univerzální generátor hodiny a rychlé univerzální vstupy/ výstupy RGPIO.

Obrázek 13. Mikrokontrolér MCF51AC256

Tento čip je použit na univerzální desce. Tedy celý prototyp robotického zařízení, tento typ mikrokontroléru používá. Pro analýzu obrazu jsou použity téměř všechny piny, a proto je vybrán typ s 80-ti piny. Používá se i ke komunikaci mezi jednotlivými částmi robotického zařízení a je tedy nutné k této skutečnosti přihlédnout a získaná data z analýzy obrazu, které je nutné odeslat, přizpůsobit možnostem čipu.

2.15.Vývojový kit pro MCF51AC256

Jelikož se kamerový senzor MT9V131připojuje k univerzální desce, která se vyrábí paralelně, je dobré nejprve desku s čipem naprogramovat na vývojovém kitu, jak z důvodu

úspory času, tak i ke zjištění, zda je deska vyrobena správně. A proto se k tomuto kitu vyrobila deska pro čip MCF51AC256.

Obrázek 14. Vývojový kit freescale

3. Návrh systému pro rozpoznání obrazu

Robotická hlava reaguje na hlasové příkazy. Tuto funkci provádí jedna z univerzálních desek používána zvukovým zařízením. Těchto desek je v hlavě více. Pro každou funkci hlavy jedna. Mezi deskami je proto vyřešen komunikační protokol. V případě, že jedna z desek není připojena, jde serva hlavy ovládat pomocí dálkového ovládání.

Zároveň komunikační protokol odesílá data přijatá z kamery. Jedná se o obraz, který je zobrazován na dálkovém ovládání a zpracovaná data jako je informace o počtu objektů, tvaru a pozici hledaných objektů.

3.1. Ovládání robotické hlavy pomocí obrazu

Robotická hlava bude mít ze strany dva mikrofony, které budou snímat okolní zvuk a reagovat na příkazy. Princip spočívá v tom, že při zavolání příkazu: Sleduj obličej nebo červený čtverec, je tato informace zpracována zvukovou deskou, vyhodnocena a po sběrnici CAN je odeslán paket dat, obsahující informaci o tom, jaký příkaz byl vysloven a pro kterou desku je určen. Pokud se jedná o obličej, tvar nebo barvu, je tato informace určena pro desku s kamerovým senzorem. V takovémto případě se prozkoumá obraz z kamery a pomocí algoritmu vyhodnotí, na jakých souřadnicích se hledaný objekt nachází.

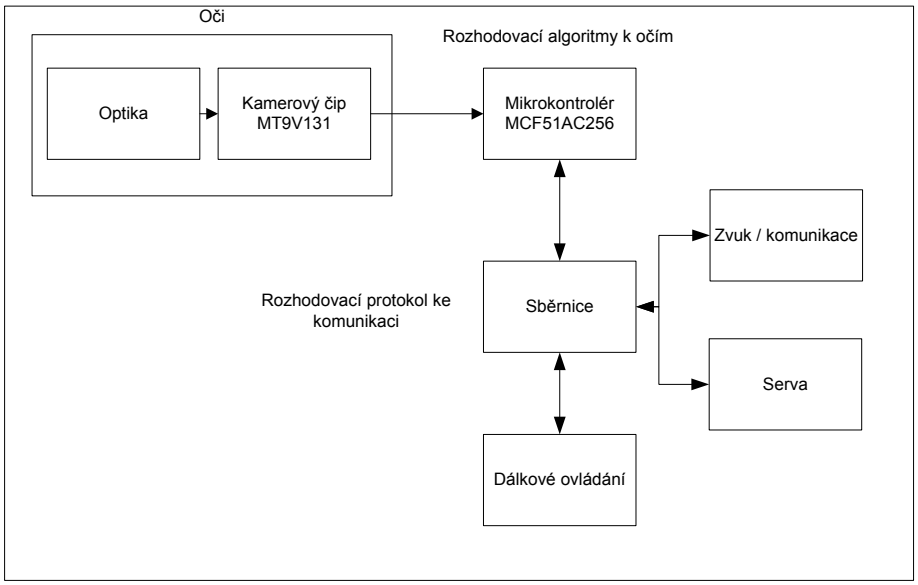
Tyto souřadnice jsou odeslány zpět na sběrnici CAN a vyhodnoceny, pro kterou desku jsou určeny. V tomto případě budou patřit desce ovládající serva hlavy. Po přijetí dat, jsou tytou informace zpracovány a přepočítány na jakou pozici se má hlava, či kamerové senzory pootočit.

Je také možné, že pro nutnost testování je nutné jisté desky vynechávat. Proto, je možnost zavolat příkaz: Otoč se na dané souřadnice, tato informace je zpracována a odeslána po CAN ihned na desku se servy. A ta otočí hlavu na požadovanou pozici.

Obrázek 15. Robotická hlava

3.2. Rozhodovací protokol

Když robot zaznamená hledaný objekt a zjistí střed, odešle informaci o poloze na sběrnici a ta odešle tyto souřadnice na servomotory umístěna na spodní části robotické hlavy a ta se pootočí tak, aby byl střed tohoto objektu ve středu obrazu. Bude tedy efekt hlavy vypadat tak, že hlava tento objekt sleduje.



Obrázek 16. Blokové schéma rozhodovacího algoritmu

3.3. Dálkové ovládání

Každou periferii v hlavě ovládá jedna z univerzálních desek. V případě, že jedna z desek nebude schopna plnit své povinnosti, je tu možnost dálkového ovládání. Tato možnost je dobrá pro vývoj a testování funkčnosti především serv, ale i dalších funkcí hlavy.

Dálkové ovládání zobrazuje snímaný obraz z kamery a informace o objektu. Z dálkového ovládání je také možné měnit nastavení, stejně u hlasového ovládání.

3.4. Odesílání dat po sběrnici

Univerzální desky uvnitř robotí hlavy společně komunikují po sběrnici. K tomuto účelu byl vybrán CAN protokol. Každá z desek zpracovává svou část. Obrazová deska zpracovává obraz z kamerového senzoru. Zvuková deska zpracovává zvuk a analyzuje lidskou řeč a pohybová deska ovládá serva hlavy a očí. Každá z těchto desek je samostatně funkční. Když mají tyto desky spolupracovat, bylo nutné vytvořit komunikační protokol.

Robotická hlava dostane příkaz. A to buďto z dálkového ovládání anebo přes zvukovou desku. Ta odešle informaci po sběrnici rozhodovacímu protokolu. Ten rozhodne pro koho je informace určena. Když se bude jednat o informace o barvě nebo tvaru, je odeslán požadavek na kamerovou desku. Ta prohledá obraz a po sběrnici se odešle informace obsahující odpověď na požadovaný příkaz. Pokud bude výsledná informace odpověď, rozhodne rozhodovací protokol o tom, aby byl příkaz odeslán zpět na zvukovou desku. Pokud bude potřeba otočit hlavou nebo očima odešle se příkaz na desku ovládající serva.

Zároveň se po sběrnici odesílá snímaný obraz. To je velký objem dat. Teoretická rychlost CAN je 1 Mb/s [13] a velikost jednoho snímku se vypočítá:

640 x 480 = 307200 Pix = 307 kPix Rovnice 10

V 8bit výstupním formátu to je

8/8 = 1 Rovnice 11

307 x 1 = 307kB Rovnice 12

A rychlost snímání je 15fps

307 x 15 = 4,608 MB/s Rovnice 13

Velikost jednoho snímku je 307kB, ale při snímkovací frekvenci 15fps je objem dat 4,608 MB/s. Takovéto množství dat nelze po CAN odesílat, proto je nezbytné odesílat upravený obraz anebo konvertovaný.

4. Návrh a realizace systému pro rozpoznání obrazu

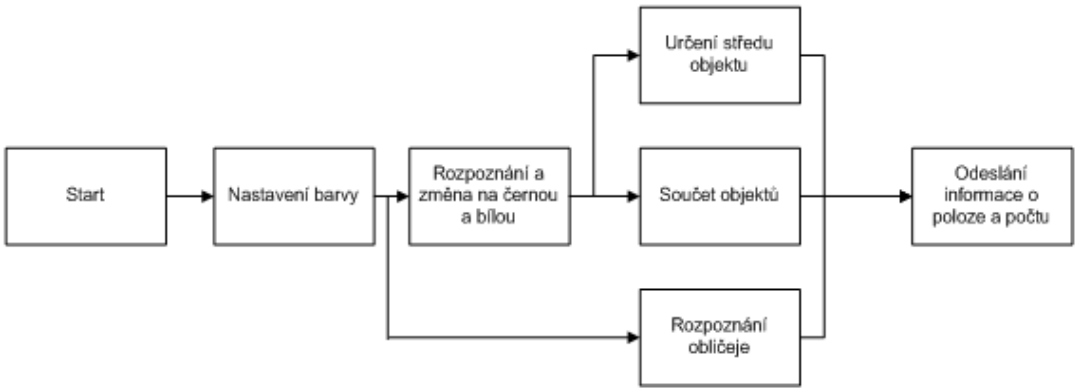
Systém pracuje na třech univerzálních deskách a Toweru. [14] Tower slouží ke komunikaci a rozhodovacímu protokolu. Další tři desky ovládají serva, zvuk a obraz. K ovládání je k dispozici ještě dálkové ovládání. Veškerá komunikace mezi těmito zařízeními probíhá po CAN komunikaci.

Kamerový senzor snímá okolní obraz ve vzdálenosti, na jakou je zaměřena optika. Sejmutý obraz je uložený v poli. Podle nastaveného zadání, jako je detekce obličeje nebo tvaru, rozloží obraz na černo bílý obraz a v případě obličeje ho rozloží na přednastavenou trikoloru. Dále se obraz podle algoritmu prohledá a obraz se zpracuje. U vyhledání obličeje nebo středu objektu odešle pouze informaci, kde se hledaný objekt nachází v podobě Xové a Yové souřadnice. V případě detekce tvaru odešle string obsahující text popisující tvar.

4.1. Graficko-uživatelské rozhraní

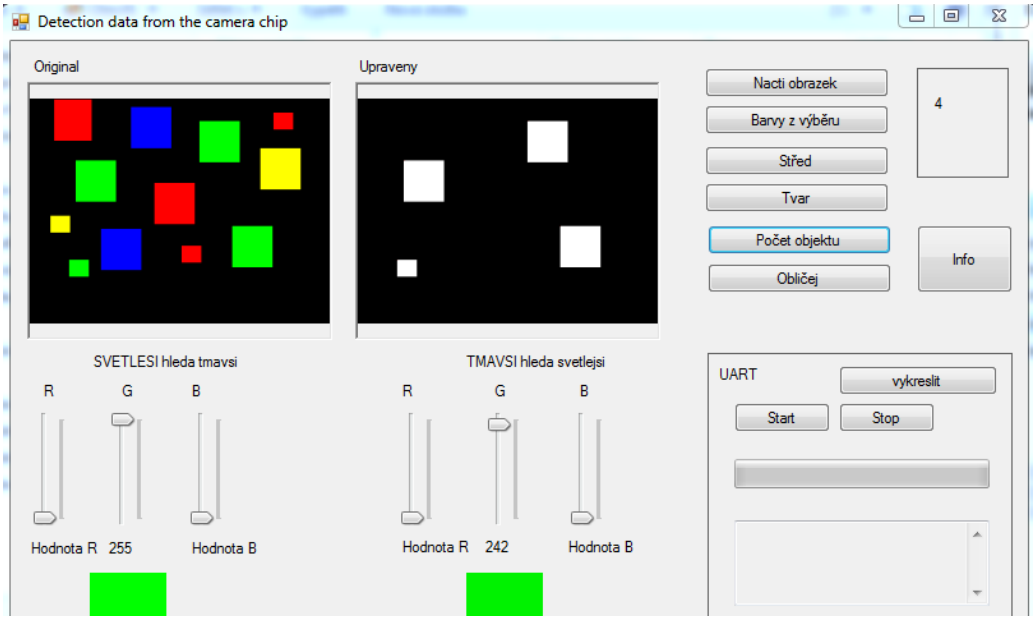
Graficko-uživatelské rozhraní je vloženo v dálkovém ovládání robotického zařízení. Robot bude sice plně autonomní a bude schopen rozpoznávat úkoly pouze ústním zadáním, ale pro přehled a testování je zde vyvinuta i tato volba.

Na blokovém schématu je vidět principielně, jak jednotlivé algoritmy zpracují obraz a jednotlivé funkce, které lze vyvolat.



Obrázek 17. Princip algoritmu zpracovávající obraz

Na GUI je na levé straně originální obraz, který chceme analyzovat. Na trackBarech nastavíme rozmezí barev, které chceme hledat a tlačítky na pravé straně vybíráme akce.



Obrázek 18. Graficko-uživatelské rozhraní

Na obrázku č. 17 je vidět příklad hledání zelených čtverců. Obrázek je nahrán z databáze, která je přiložená na CD. Ale kdyby senzor snímal barevné obrazy, tento obraz by se objevil při stlačení tlačítka start a zaměřením kamery na obraz. V pravém horním rohu je ve čtverci vidět odpověď 4, která by se odeslala po sběrnici.

4.2. Komunikace po sběrnici CAN

Veškerá komunikace mezi jednotlivými deskami je prováděna po sběrnici CAN. Robotická hlava obsahuje tři univerzální desky. Ty ovládají zvuk, sluch nebo pohyb hlavy pomocí servo pohonu, dále tower s komunikací, a deska s obrazem.

Princip:
Na desku s kamerovým senzorem, je odeslán požadavek na zjištění středu objektu. Algoritmus vypočítá souřadnice X a Y. Ty jsou odeslány po sběrnici.

mech_data.mouth.position.vertical = 500;

Algoritmus vytvořený k odesílání dat nejprve odešle informaci o pozici a odešle data po sběrnici následujícím kódem.

CAN1_SendFrame(0,CAN_MSG_ID_POSITION,DATA_FRAME,8,data);

Hlava se otočí na požadovanou pozici. A je možné zadat další příkaz, nebo nechat hlavu stále sledovat a tím způsobit efekt otáčení se hlavy za objektem.

Je třeba počítat s tím, že na hlavě se nachází dvě serva. Jedno ovládá hlavu a jedno oči.

```
data[1] = (uint8_t)mech_data.eye.position.horizontal;
data[2] = (uint8_t)mech_data.eye.position.vertical;
data[3] = (uint8_t)mech_data.head.position.horizontal;
data[4] = (uint8_t)mech_data.head.position.vertical;
```

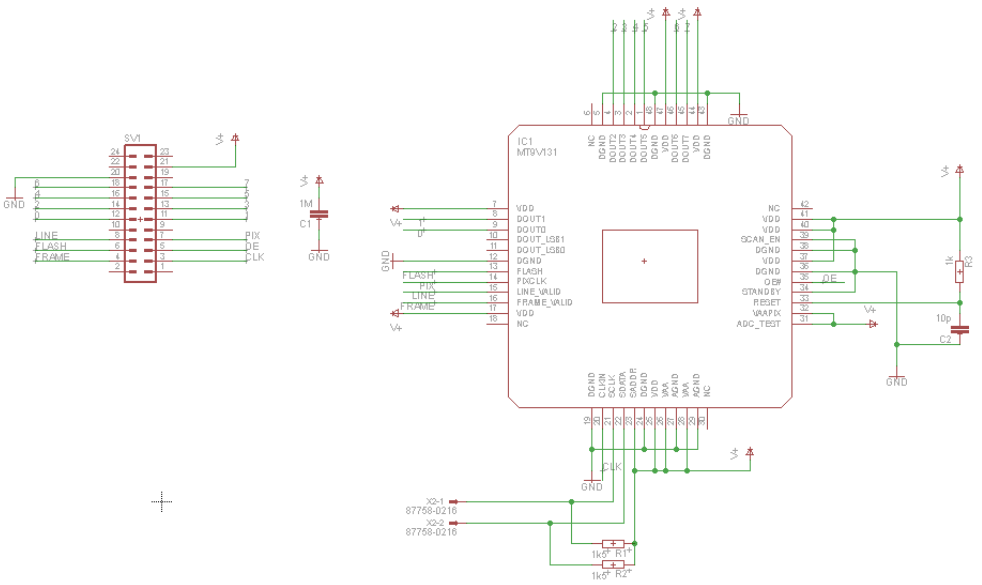
Tento kód popisuje pouze ovládání serv, ale po CAN se ovládá celá hlava. Tedy i zvuková deska a veškerá komunikace mezi periferiemi. Princip odesílání dat je stejný a celý kód je v příloze.

5. Hardwarové řešení kamerového senzoru

Je vyrobena univerzální deska s čipem MCF51AC256. K této desce je vyrobena odnímatelná část, nesoucí samostatný kamerový senzor a optiku. Tato deska plošných spojů je vyrobena tak, aby jí bylo možné vložit do hlavy robotického zařízení a pohybovat s ním. Schéma i deska plošných spojů jsou navrhnuty v programu Eagle.

5.1. Schéma

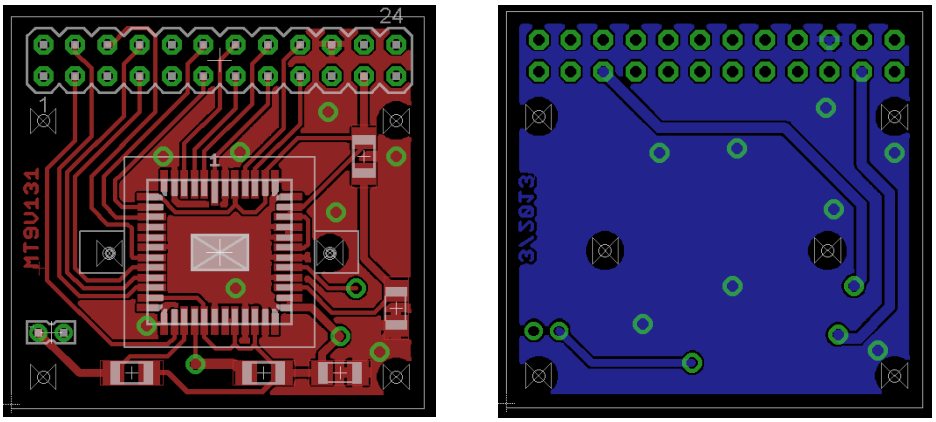
Samotné schéma je tvořeno senzorem MT9V131. K tomuto senzoru jsou připojeny nezbytné součástky k jeho bezchybnému provozu. Hodnoty těchto součástek jsou vyčteny z datasheetu k tomuto senzoru [1]. Toto schéma je navrženo tak, aby v případě potřeby bylo možné, použít jakýkoli pin. Také byly vyvedeny programovatelné piny SCL a SDA. Ty jsou připojeny na samostatnou patici. Ostatní piny jsou připojeny na společnou patici, která je připojena k univerzální desce. Celé schéma je v příloze.



Obrázek 19. Schéma zapojení kamerového senzoru

5.2. Deska plošných spojů

Návrh je vyroben na oboustrannou desku. V desce jsou čtyři vrty na připojení k pohybovému mechanizmu robotického zařízení a dva vrty na připevnění optiky k desce. Deska má rozměry 30 x 35mm.

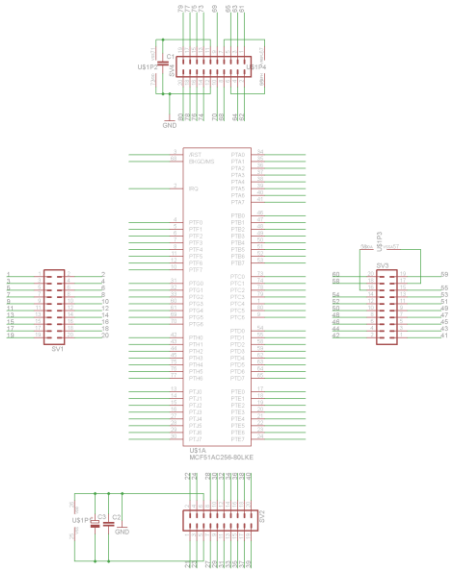


Obrázek 20. DPS kamerového senzoru (z leva strana TOP a BOTTOM)

5.3. Schéma a DPS externí desky k vývojovému kytu.

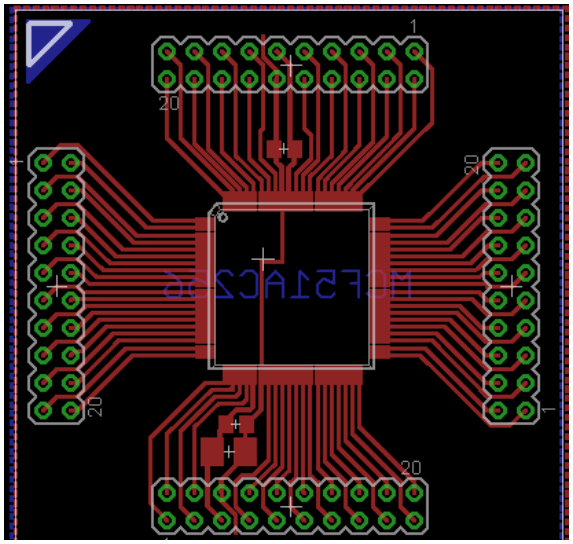
Deska se vyráběla z potřeby programovat části robota v době, kdy ještě nebyla vyrobena hlavní deska.

Na desce se nachází vývody pro jednotlivé piny a součástky potřebné k napájení procesoru. Ty byly vybrány podle technické dokumentace od výrobce [2]. Celé schéma je v příloze



Obrázek 21. Schéma desky pro vývojový kit

Deska byla vyráběna na oboustrannou desku podle vzoru jiné desky s jiným procesorem, která byla určena pro tento vývojový kit. V levém horním rohu je značka, určující směr zapojení desky na kit, aby nedošlo k přepólování.



Obrázek 22. Deska plošných spojů pro vývojový kit

6. Metody pro analýzu obrazového signálu s detekcí barev, tvarů a lidských obličejů.

Ve třetí kapitole je popsáno, jak jednotlivé algoritmy pracují. Od detekce barev a jejich změny na černou a bílou, přes spočítání objektů v obraze, tak až po objevení lidského obličeje. Algoritmy jsou popsány UML diagramy a je zde popsán kód v jazyce C. Změny těchto nastavení, lze nastavit buď na dálkovém ovládání anebo hlasem.

Základem detekce je rozpoznávání barev. Kamerový senzor odesílá obraz v YUV formátu a kód je vytvořený pro RGB. Proto je nezbytné po načtení obrazu, celý image pomocí vzorců přepočítat a překreslit.

Pokud je detekce barev vyřešená, je hledání v obraze, už jen o porovnávání předchozích a následujících pixelů. Každá detekce má vlastní algoritmus. Při detekci tvaru hledám pouze černou a bílou barvu. Není tedy nezbytné mít barevnou kameru. Nebo načítat pouze jasovou složku. Pro detekci obličeje je však barva potřeba.

6.1. Výběr a detekce požadované barvy

Základem detekce je bezpochyby detekce barvy. Od této schopnosti se odvíjí veškeré další operace.

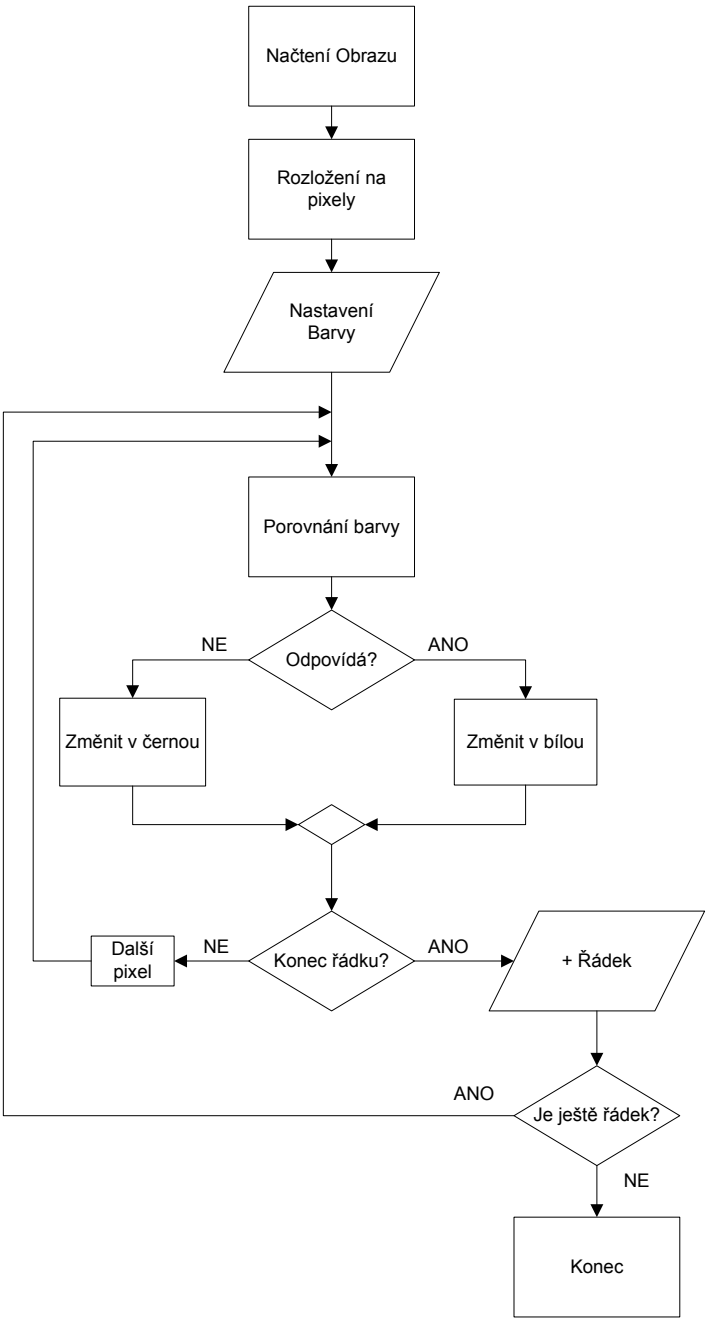
Obraz z kamerového senzoru je snímán v dané sekvenci, tak jak je popsáno v kapitole 2.21 čtecí sekvence a synchronizace pro kamerový senzor MT9V131. Jelikož jsou používané detekce barev, budu snímat složky U a V. Ty, se pomocí vzorců z kapitoly: Přepočet ze složek YUV na RGB, změní a je možné z nich pomocí těchto tří složek vyhledat požadovanou barvu. Když je obraz překreslen, je možné nastavit hledanou barvu. Vždy je nastaveno rozmezí barev. To znamená, že ve vyhledávání není 16milionů barev, ale omezené množství, podle toho, jak přesné hledání je prováděno. Zároveň záleží na osvětlení. Pokud je v místnosti šero nebo tma budou se hledat tmavší pixely. Pokud naopak světlo budou to světlé pixely. Proto je v této aplikaci každá barva definována tak ze široka.

6.1.1. UML diagram pro detekci barvy

V UML diagramu na obrázku č. 19, je znázorněno, jak se mění jednotlivé barevné pixely načtené ze zdroje a mění se podle nastavení na černou a bílou.

V GUI jsou nastavovací prvky v podobě šesti trackBarů, které nastaví rozmezí barev, které jsou nutné k vyhledání. Tyto nastavené barvy a jejich se pro zjednodušení zobrazují v panelech. Pokud se barva nachází v nastaveném rozmezí, změní se v bílou. Pokud ne, změní se v černou. Takto dosáhneme černobílého obrazu.

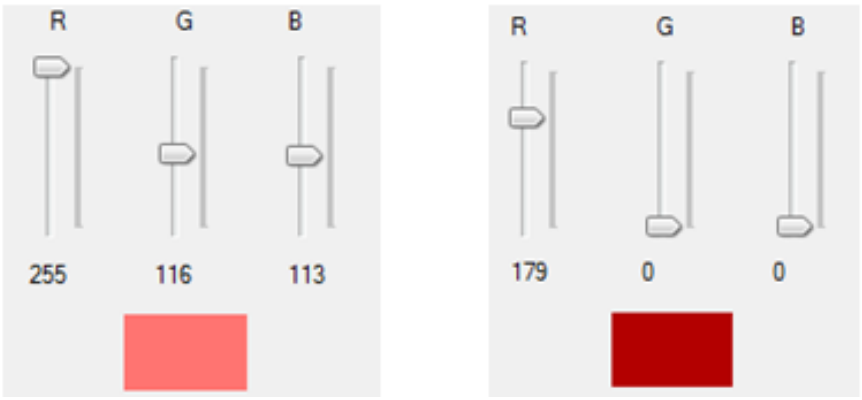
Tato funkce je vhodná jako základní kontrola při testování, nastavování a synchronizaci robotického zařízení. Například zda objeví správnou barvu ze sledovaného objektu.



Obrázek 23. UML diagram výběru a změny barvy

6.1.2. **Nastavení v GUI**

Po načtení obrazu do proměnné se obraz rozloží do R, G, B v každém pixelu. Pomocí šesti trackBarů se nastaví požadovaná barva. Třemi se nastaví světlejší a třemi tmavší barva. Všechny barvy mezi těmito nastavenými se vyhledají.



Obrázek 24. Prvky k nastavení barvy

Každý jeden trackBar nastavuje jednu barvu R, G nebo B. Nyní nastává porovnávání barev. Pokud je nastavená barva detekována změní se v bílou barvu, vše ostatní se změní v černou.

Obrázek 25. Změna barev po nastavení

Tímto způsobem je přistoupeno ke každému pixelu v obraze zvlášť. Nejprve se definuje c jako strukturovaná proměnná. První *for* přistoupí ke každému pixelu v prvním řádku, dokud nedorazí na konec. Poté přijde na řadu druhý *for*, který se posune na další řádek a situace se opakuje, dokud se neprohledá celý obraz.

6.1.3. **Program v jazyce C pro detekci barvy**

Dva do sebe vnořené *fory* umožní prohledat postupně každý pixel. Ve Visual studiu je možné použít funkci *Color*, která rozkládá pixel na čtyři jednotlivé složky v hodnotě 0-255. Těmi čtyřmi složkami je R- červená, G- zelená, B- modrá, A- jas.

```
Color c;
for (int i = 0; i < sourceimage.Width; i++)
{
      for (int j = 0; j < sourceimage.Height; j++)
      {

      }
}
return sourceimage;
```

Funkce *color* nejprve rozloží každý pixel podle pozice prohledávání a proměnných i aj. Ve funkci *color*, je použita metoda *FromArgb*. Ta určí barvu pixelu, na který je zrovna přistupováno, na hodnotu, která je nastavena. Barvu lze nastavit jako RGB hodnotu. Každá hodnota je v rozmezí 0 – 255. Když budou všechny 3 hodnoty nastaveny na 255, bude výsledná barva bílá. Pokud to budou 0, tak černá. Poslední řádek nastaví tuto novou barvu do příslušného pixelu. Tento kód se vkládá mezi závorky druhého *foru*.

```
c = sourceimage.GetPixel(i, j);
c = Color.FromArgb(255, 255, 255);
sourceimage.SetPixel(i, j, c);
```

Bylo nezbytné, vytvořit komplexní podmínku, pouze pro vybranou barvu.

```
if (c.R <= Class.R && c.G <= Class.G && c.B <= Class.B)
```

Byly vytvořeny 3 trackBary, které měly rozmezí od 0 do 255. Ty tuto hodnotu uloží do proměnných R, G, B ve třídě *Class*. Tyto hodnoty byly porovnávány touto podmínkou s hodnotami ve funkci *color*. Pokud byla barva pixelu tmavší než zadaná, byla vybrána.

```
if (c.R >= Class.r && c.G >= Class.g && c.B >= Class.b)
```

Pro dokončení vyhledání rozmezí barvy, bylo potřeba vytvořit druhou podmínku a tím i další 3 trackBary a 3 proměnné r, g, b a tyto hodnoty byly opět porovnány s hodnotami ve funkci *color*. Pokud byla barva pixelu světlejší než zadaná, byla vybrána.

Tímto způsobem je docíleno výběru barvy v nastavitelném rozmezí. Když je nalezena barva, která je hledaná, tak se změní v bílou. Ostatní barvy jsou změněny na černou.

6.2. Vyhledání středu vybraného objektu

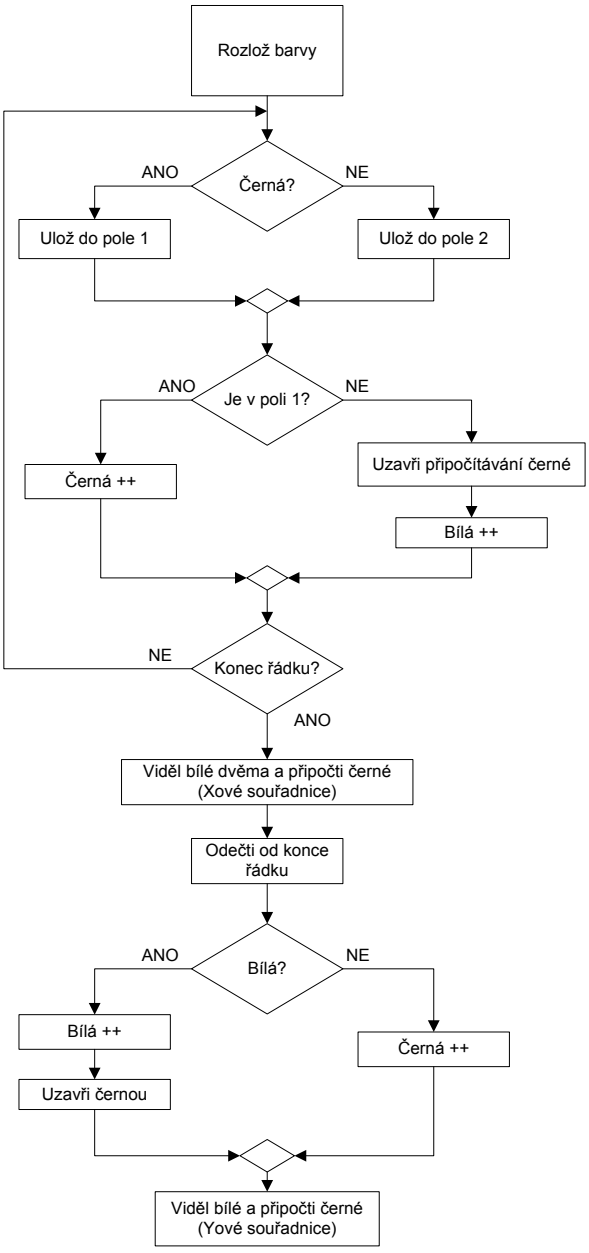
Po rozdělení obrazu na dvě barvy a to černou a bílou, je možné vypočítat střed zobrazeného tvaru.

Každý řádek zvlášť ukládá algoritmus do pole hodnoty 1 a 2. Pokud je pole černé uloží se 1, pokud bílé uloží se 2. Když je pole černé, do proměnné *cerna* se přičítá hodnota 1. Jakmile se objeví bílé pole, začne se přidávat hodnota 1 do proměnné *bila* a zakáže se připočítávat další

černé pixely. Když se vezme hodnota proměnné *bila* a vydělí se dvěma a k ní se připočítá hodnota černé, vznikne nová proměnná *jedna*.

Takto se prohledá celý bílý objekt. Počet bílých řádků se vypočítá tak, že ze šířky obrazu se odečte jedna a pokud je výsledek menší než šířka obrazu, tak se na řádku objekt objevuje a připočítá se řádek. Takto se spočítají černé řádky, bílé řádky. Při připočítávání bílých řádků se zakáže opětovné počítání černých řádků jako v případě počítání *X-ové* souřadnice. Opět se bílé sloupce vydělí dvěma a připočítají se černé sloupce. Tak dostaneme *Y-ovou* souřadnici.

6.2.1. **UML diagram pro vypočítání středu vybraného objektu**



Obrázek 26. UML diagram k výpočtu středu objektu

Po změně barev na černobílou, je nalezen objekt o zatím nedefinovaných rozměrech. Pokud chceme, aby se robotí hlava otočila ve směru hledaného objektu, je nutné zjistit souřadnice středu.

Program funguje tak, že při detekci pixelu na řádku, do jednoho pole připočítávám černé a do druhého bílé. Jakmile narazím na bílou, tak další černou nepočítám. Bílé pixely program vydělí dvěma a připočítá černé. Takto máme *Xovou* souřadnici. Po objevení bílé program počítá řádky a na konci je princip stejný sečte se polovina bílých a všechny černé pixely. Takto se vypočítá *Yova* souřadnice.

Tyto nalezené souřadnice jsou odeslány po CAN sběrnici jako střed objektu. Ty jsou zpracovány a robotická hlava se otočí tak, aby byl střed objektu uprostřed snímaného obrazu. Po dokončení akce se může provést kontrolní test, zda je objekt skutečně uprostřed.

6.2.2. **Program v jazyce C pro vypočítání středu vybraného objekt**

Vyhledání středu je vyvinuto tak, že obraz je prohledá stejným algoritmem, jako při změně barvy. Však nyní hledá pouze bílé barvy.

Následně je ve stejném cyklu vyhledáno a porovnáno, do kterého pole se proměnná uložila. Pro urychlení nebo přehlednost by se tato funkce dala vložit do jedné avšak pro další rozšíření a složitější výpočty je třeba mít tato data v poli.

```
// pole
if (pole[l] == 2 )                    //bile
                {
                    dve = 1;
                    bile++;
                }
if (pole[l] == 1 && dve == 0)         //cerne
                {
                    cerne++;
                }
```

Po dosažení konce řádku jsou všechna data, potřebná k výpočtu vynulována a v dalším cyklu opět použita.

Když je detekovaná černá barva, do pole s názvem *pole* je uložena pod pozicí 1. Pokud je detekována bílá barva je do tohoto pole vložena 2.

Pokud je v poli *pole* přičtena hodnota 1, což je černá, počítá se do proměnné *cerne* o jedna víc. Když je přičtena hodnota 2, což je bílá změní se proměnná *dve* na hodnotu 1. Tím je docíleno, aby se další černé pixely nepřipočítávaly.

Funkce graf je zavolána pouze jednou za řádek. Zde je vypočítána *Xová* a *Yová* souřadnice středu bílého tvaru.

```
//Xove souradnice
jedna = (bile / 2) + cerne;
```

Všechny bílé pixely jsou vyděleny dvěma a přičteny ke všem černým pixelům na levé straně. Takto je vypočítán střed bílého tvaru na řádku. Aby bylo možné vypočítat, kolik řádků má bílý tvar, byl vyvinut algoritmus na porovnání, zdali bylo dosaženo konce tvaru či nikoli.

```
//Yove souradnice
Yova = (PomBilaY / 2) + YovaCerna;
```

Pokud je celý řádek černý, není tedy na řádku detekován hledaný tvar je do proměnné *YovaCerna* připočtena o jedna víc. Pokud je tvar a tedy i bílá barva objevena je zabráněno počítat další černé řádky a počítají se už pouze bílé. Ty se pak vydělí dvěma a přičteny k černým.

Takto jsou vypočítány *X* a *Y* souřadnice středu bílého tvaru.

6.3. Počet objektů v obraze

V případě, že se za objektivem objeví více objektů, algoritmus by detekoval vždy jen ten nejvíce vlevo nahoře jakou souřadnici 0,0. Proto byl vyvinut algoritmus, pro zjištění počtu objektů, tak aby při příkazu například, kolik zelených objektů se zde nachází, bylo možné odpovědět. Po nastavení požadované barvy, je tato barva změněna na bílou. V případě pole na číslo 1. Zbytek pole se změní na černou anebo na číslo 2. V těchto polích se následně vyhledávají tyto položky a podle algoritmu detekují jednotlivé objekty.

Princip spočívá v prohledávání jednotlivých sloupců a hledání objektů. Pokud je objeven objekt, je položena otázka, zda je to nový objekt či byl předcházející akcí, už objeven. To se provádí pomocí série podmínek. Pokud se jedná o nový objekt je připočítána do proměnné jedna na víc.

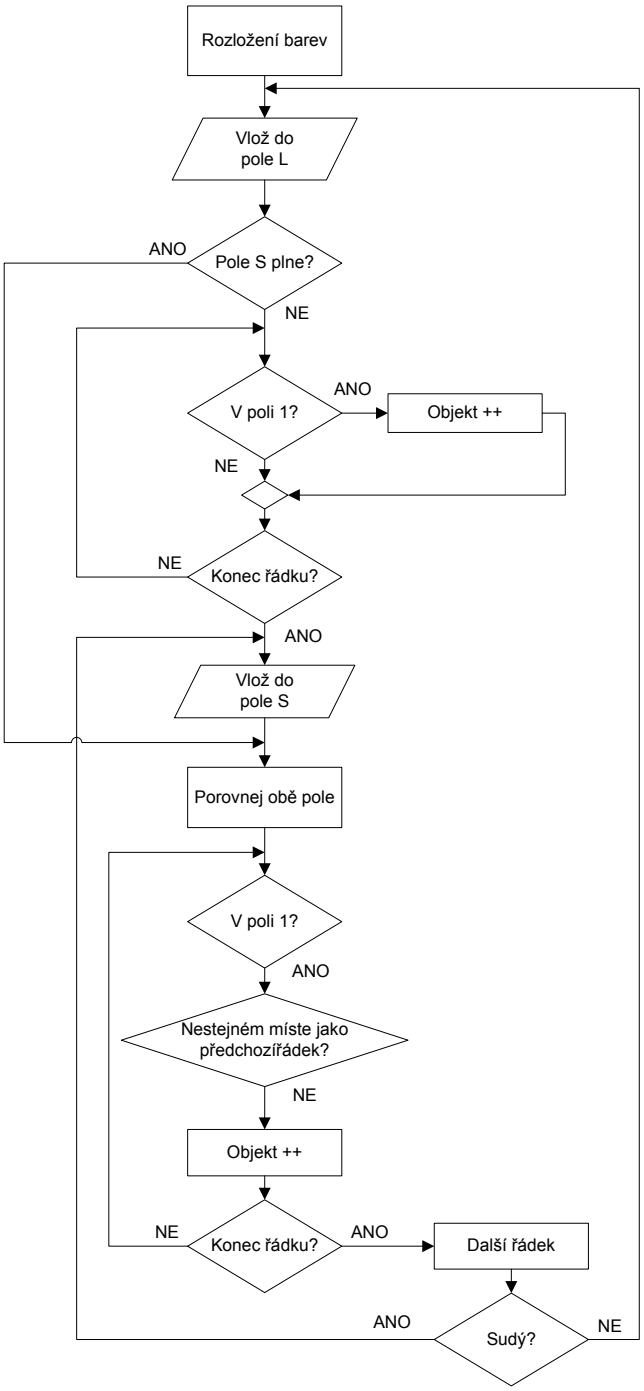
6.3.1. UML diagram pro zjištění počtu objektů v obraze

Vždy se porovnává liché a sudé pole, což jsou sloupce obrazu a zjišťuje se, zda objekty patří k sobě nebo zda se jedná o nový objekt.

Celé první pole, to je liché pole se uloží do pole označeného jako *L*. Pokud je pole S prázdné začne se v prvním řádku hledat objekt. Pokud se nějaký objeví tak se do proměnné objekt připočítá hodnota jedna.

Druhý řádek se taktéž uloží do pole. Tentokrát do sudého pole označeného *S*. Pole se prohledá, a pokud je v něm objeven objekt, porovná se s předchozím řádkem. Pokud je výsledkem porovnávání, že na předchozím řádku byl objekt na sejném místě, nic se neděje. Pokud je na předchozím řádku černé místo je to nový objekt a do proměnné objekt se připočítá hodnota 1.

Na konci algoritmu se rozhodne, zda následující řádek bude lichý nebo sudý a ten se uloží do sudého nebo lichého pole a začne se opět porovnávat s předchozím řádkem.



Obrázek 27. UML diagram k součtu objektů v obraze

6.3.2. **Program v jazyce C pro počet objektů v obraze**

Důležitou podmínkou k tomuto algoritmu je, aby bylo možné rozpoznat lichý a sudý řádek. Je tedy na konci každého řádku přičtena hodnota 1 do proměnné *radek* a při určování sudého nebo lichého řádku se používá tato podmínka.

```
if (radek % 2 == 0)
```

Když je rozhodnuto, zda je sudý nebo lichý řádek na řadě porovná se černá a bíla barva a uloží se do pole jako 1 nebo 0 pod aktuální pozicí, kterou přičítám při každém pixelu.

Pole je nadále porovnáno podmínkami, zda je možné ho připočítat jako nový objekt anebo, zdali se nejedná o stejný objekt.

```
if (poleLiche[pozice] == 1)
                    {
                        if (nepustit == 0)
                    {
if (poleLiche[x] < poleSude[x])
                    {
                        pocetObjektuLiche++;
                        nepustit = 1;
                    }
```

Sudý řádek se počítá obdobně jako lichý s výjimkou podmínky a ke každé proměnné je přiřazena hodnota 2.

Když jsou obě pole plná, opět se porovnávají a pomocí sady podmínek se rozhodne, zda byl objekt již objeven, nebo zda se jedná o nový objekt a možné ho přičíst.

```
if (poleLiche[v] < poleSude[v])
                   {
                        if (nepustit2 == 0)
                   {
                        pocetObjektuSude++;
                        nepustit2 = 1;
                   }
```

Když je rozhodnuto, jestli objekt připočíst nebo ne, jsou proměnné vynulovány, aby bylo možné vložit nový řádek do pole. Proměnné jsou vždy nulovány tak, že když jsem v sudém poli, nuluje se liché pole a obráceně. Proměnné *nepustit* a *nepustit2* se nulují pokaždé.

```
if (radek % 2 == 0)
        {
            pozice = 0;
            pocetObjektuLiche = 0;
            poleLiche = null;
        }
else
        {
            pozice2 = 0;
            pocetObjektuSude = 0;
            poleSude = null;
        }
        nepustit = 0;
        nepustit2 = 0;
```

6.4. Tvar

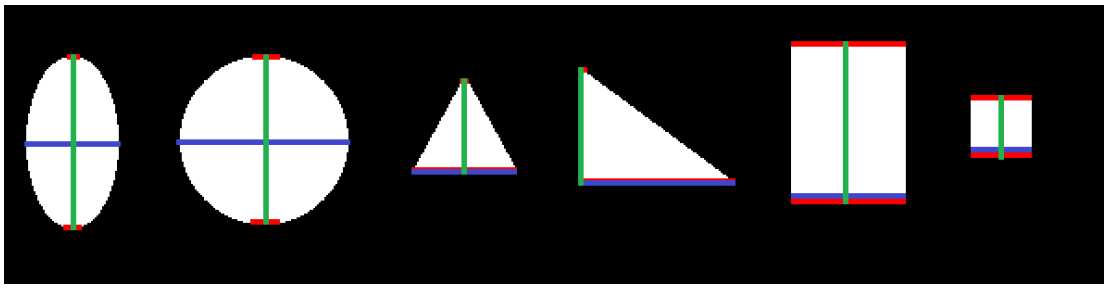
Pro robotické zařízení je dobré, když umí rozpoznávat tvary, ať už se dá použít k rozpoznávání značek, nebo součtu stejných objektů v obraze. Následující řešení se dá použít do detekce až šestiúhelníku, což je maximální počet úhlů objevujících se ve volné přírodě[11].

Při načtení tvaru v nelaboratorních podmínkách bude mít za následek, že na každé straně objektu bude mít barva jiný jas. Je tedy nezbytné nastavit barvy hledaného objektu tak, aby našel celý objekt nejen část. To lze vyřešit širším nastavením barev.

6.4.1. Princip detekce tvaru

K detekci je potřeba jen čtyř hodnot a to: počet pixelů na prvním řádku a počet pixelů na posledním řádku. Tyto hodnoty jsou na obrázku označeny červeně. Dále výšku objektu. Ta je značena zeleně a pak maximální šířku v objektu a ta je zobrazena modře. Pro názornost se u trojúhelníků a čtverce s obdélníkem ve spodní části překrývají barvy jen z části. V programu by měly tyto hodnoty stejnou velikost.

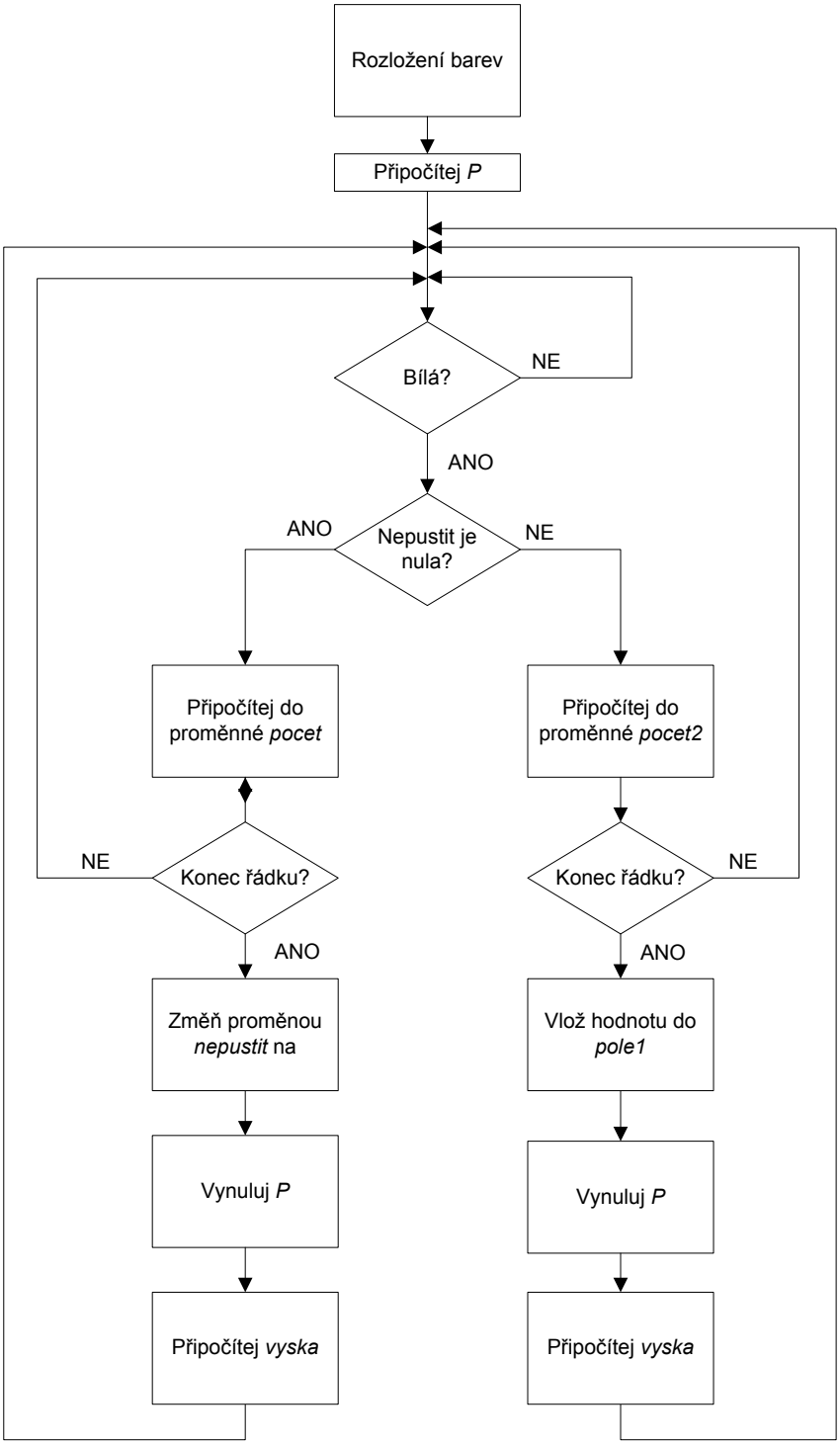
Po nastavení jakou barvu má objekt mít a změně v černobílý obaz začíná algoritmus hledat bílé pixely. Při detekci bílé barvy načte počet bílých pixelů na řádku. Počká na konec řádku a hledá dál. Pi detekci další začne ukládat několik hodnot. Jedna z nich je přepisovatelná proměnná, která hledá nejvyšší dosažené číslo. Další měří sloupec. Tedy nalezne-li se na řádku bílý pixel, je na řádku přičtena jedna proměnná. Čtvrtá a poslední proměnná se neustále přepisuje, dokud nenarazí na poslední řádek. Tím je umožněno detekovat objekt.



Obrázek 28. Příklady obrázků a detekce hodnot pro jejich identifikaci

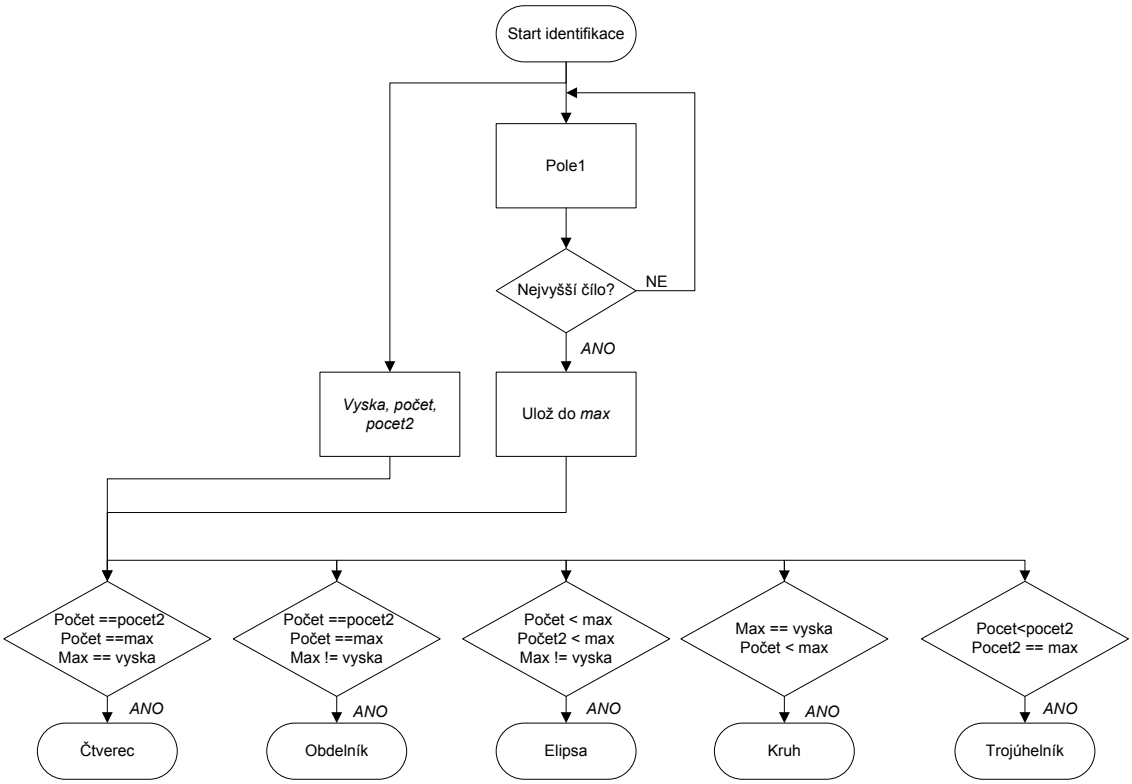
6.4.2. UML diagram pro detekci tvaru

Algoritmus nejprve hledá bílou barvu. Když ji objeví, uloží si, že již našel objekt a další hledat nebude. Počítá, kolik pixelů je na prvním řádku. Zároveň počítá kolik řádků má objekt, a maximální počet pixelů. Tato funkce pracuje tak, že se proměnná přepisuje, když narazí na vyšší číslo. A poslední je pole, které zaznamenává poslední řádek.



Obrázek 29. UML diagram určení tvaru

Další UML diagram je již rozhodovací algoritmus, který se sesbíraných dat rozhodne, o jaký tvar se jedná. V průběhu dalšího vývoje, by bylo možné rozhodovací algoritmus rozšířit i o další tvary, avšak pro účely tohoto robotického zařízení těchto pět tvarů dostačující.



Obrázek 30. UML diagram k rozhodnutí o tvaru

6.4.3. **Program v jazyce C pro detekci tvaru**

Nejprve celý obraz prohledáme dvěma vnořenými *fory*. Obraz je už překlopený na černobílý a není tedy nutné hledat jinou barvu než bílou, tak jak u předchozích akcí.

Po nalezení bílé následující podmínkou, se uloží do proměnné *počet* bílých pixelů na řádku.

```
if (nepustit == 0)                          //prvni radek
{
      pocet++;
      c = Color.FromArgb(255, 0, 0);
      sourceimage.SetPixel(i, j, c);
}
```

Na konci řádku se přepíše proměnná *nepustit*, jak je popsáno níže a tím se zabrání přepsání proměnné počet. Další řádky se počítají do proměnné *pocet2*, tak dlouho než se spočítá předposlední řádek a ten zůstane uložený.

```
if (nepustit == 1)          // ostatni radky(prepisuji se)
{
      pocet2++;
      c = Color.FromArgb(255, 255, 0);
```

```
                sourceimage.SetPixel(i, j, c);
            }
```

Když se narazí na konec řádku a byla na řádku detekována bílá barva, změní se proměnná *nepustit*, jak již bylo řečeno, dále se připočítá proměnná *výška* a tím se zjišťuje, na kolik řádků se detekovaný objekt rozkládá. Zároveň se uloží do pole, abychom posléze zjistily největší prvek objevený v objektu

Na konci každého řádku, se proměnná *p* vynuluje, aby bylo možné počítat na novém od začátku.

Takovýmto způsobem získáme čtyři hodnoty, které jsou nezbytné k identifikování předmětu. Jsou to velikost prvního řádku, velikost posledního řádku, velikost největšího řádku, a výšku objektu.

Vypsání třech hodnot. Výšky, prvního řádku a posledního řádku. K identifikování je nezbytné mít ještě čtvrtý rozměr a to neširšší část objektu. K tomu se požívá následující kód, který nalezne největší hodnotu v poli.

```
int id;
for (int i = 0; i < 315; i++)
            {
                  if (pole1[i] > max)
                  {
                        max = pole1[i];
                        id = i;
                  }
            }
```

Když již máme všechny čtyři hodnoty, lze pomocí sady rozhodovacích podmínek určit, o jaký předmět se jedná.

Když bude horní část stejně velká jako spodní, zároveň horní část stejně velká jako maximální a zároveň jako výška jedná se o čtverec.

```
      if(pocet == pocet2 && pocet == max && max == vyska) //ctverec
            {
                  Console.Write("Ctverec");
            }
```

Když bude horní část stejně velká jako spodní, zároveň horní stejná jako maximální, ale maximální nebude stejně velká jako výška, jedná se o obdélník

Když bude horní část menší než maximální, zároveň maximální bude větší než spodní část a maximální nebude stejná, jako výška jedná se o elipsu.

Když bude maximální hodnota stejná jako výška a zároveň bude horní část menší, než maximální jedná se o kruh

Když bude horní část menší než spodní a zároveň spodní stejně velká jako maximální jedná se o trojúhelník

6.5. Detekce obličeje

Detekce obličeje je dnes velmi často používanou funkcí v různých embedded systémech jako jsou digitální fotoaparáty. Tento návrh je vytvořen trochu odlišným způsobem, než klasické zobrazovací zařízení. Je zde otestován nový způsob detekce lidského obličeje.

6.5.1. Princip detekce obličeje

Nejprve se obraz převede na tři základní barvy. Červenou, žlutou a okrovou. Tyto barvy vzniknou na základě rozmezí nastavených barev. Program se snaží detekovat obličej, proto se barvy přibližné barvy jako je lidská tvář převede do okrové barvy, stejným způsobem se změní i rty a jim přibližné barvy na červenou a oči tedy bílá barva až šedá na žlutou.

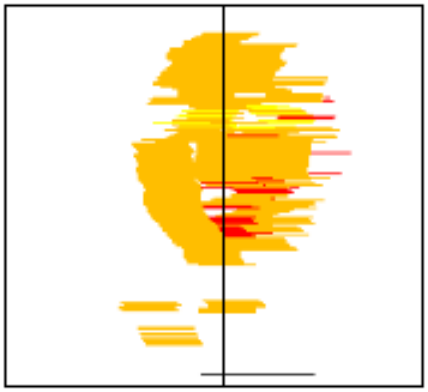
Vznikne podobný obraz jako na obrázku č. 31.

Obrázek 31. Detekce obličeje a převod na 3 barvy

Při nízkém rozlišení obrazu vzniká problém. Oči a rty jsou velmi špatně detekovatelné a proto byl navržen algoritmus tak, aby při zápisu „rozmazal“ obraz směrem doprava o tolik pixelů, kolik se do ovládače zadá. Pro představu je na č. 32 příklad rozmazání o 30 pixelů.

Obrázek 32. Detekce obličeje a převod na tři barvy s "rozmazáním"

U takto upraveného obrazu, už je možné určit, zda se jedná o obličej nebo ne. Princip této detekce spočívá v tom, že barev podobných jako se nachází na lidském obličeji, se v přírodě nachází mnoho, ale nemnoho poskládaných v tomto pořadí pod sebou. Takže při hledání obličeje byl vytvořen algoritmus, který prohlédne každý sloupec zvlášť a sleduje, zda se v jedné řadě nachází všechny tři barvy. Pokud ano, uloží pozici a zkusí sloupec vedle, pokud ano, tak opět uloží a tak dále, dokud neprohlédne celý obraz. Takto se dá zjistit s velkou přesností, kde se nachází obličej.



Obrázek 33. Objevení obličeje

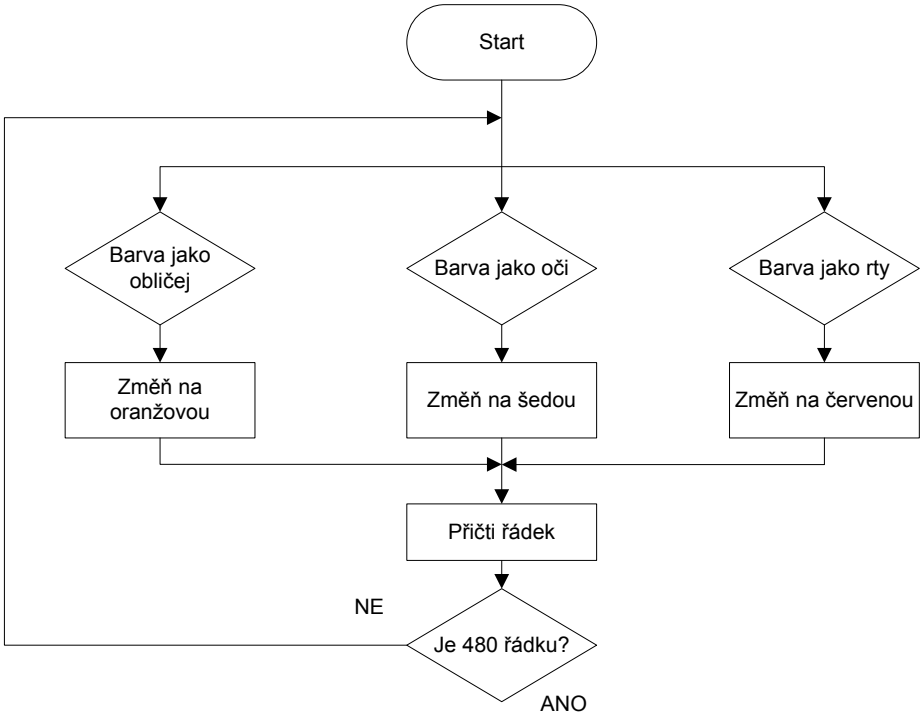
Pokud v celém obraze objeví jeden objekt, tak ze souřadnic, které si uložil, vypočítá polovinu a odečte posunutí, které bylo na začátku kvůli detekci vytvořeno. Takto získá *Xovou* souřadnici obličeje.

Yová souřadnice se vypočítá tak, že při detekci všech barev v jednom sloupci, uloží číselnou pozici bodu, kde je předpoklad na oči a od nich počítá, dokud nenarazí na bílou, tedy mimo obličej. Spočítá bílé pixely a k nim přičte polovinu obličeje. Takto vznikne *Yová* souřadnice. Princip je podobný jako u vypočítání středu objektu popsaný v kapitole 5.2.

6.5.2. **Detekce barev na lidském obličeji UML diagram**

Detekce na lidském obličeji se provádí pomocí rozhodovacího algoritmu. Obličej každého člověka má jiný odstín anebo je jiné osvětlení při snímání obrazu. Proto, je vytvořena podmínka, která vybere předpokládanou barvu v rozmezí od světlejší po tmavší. Stejným způsobem jsou vybrány i rty a oči.

Pro zjednodušení a zrychlení výpočtu jsou tyto tři barvy na obličeji přebarveny na tři unikátní barvy, se kterými se poté pracuje.



Obrázek 34. UML detekce barev na lidském obličeji

6.5.3. **Detekce barev na lidském obličeji kód v jazyce C**

Lidský obličej se skládá z unikátních barev a každý člověk má odlišně barevný obličej. Proto je vytvořen algoritmus, který detekuje standardní obličej středoevropana. Pro vývoj aplikace jsou vybrány tyto hodnoty RGB obrazu a převedeny do čistě daných barev, které se v přírodě běžně neobjevují.

Podmínka, která konvertuje barvy přibližně lidskému obličeji do oranžové.

```
//oblicej
if ((c.R <= 184 && c.G <= 134 && c.B <= 107) && (c.R >= 113 && c.G >=
71 && c.B >= 49))
```

Podmínka, která konvertuje barvy blízké lidskému oku do žluté.

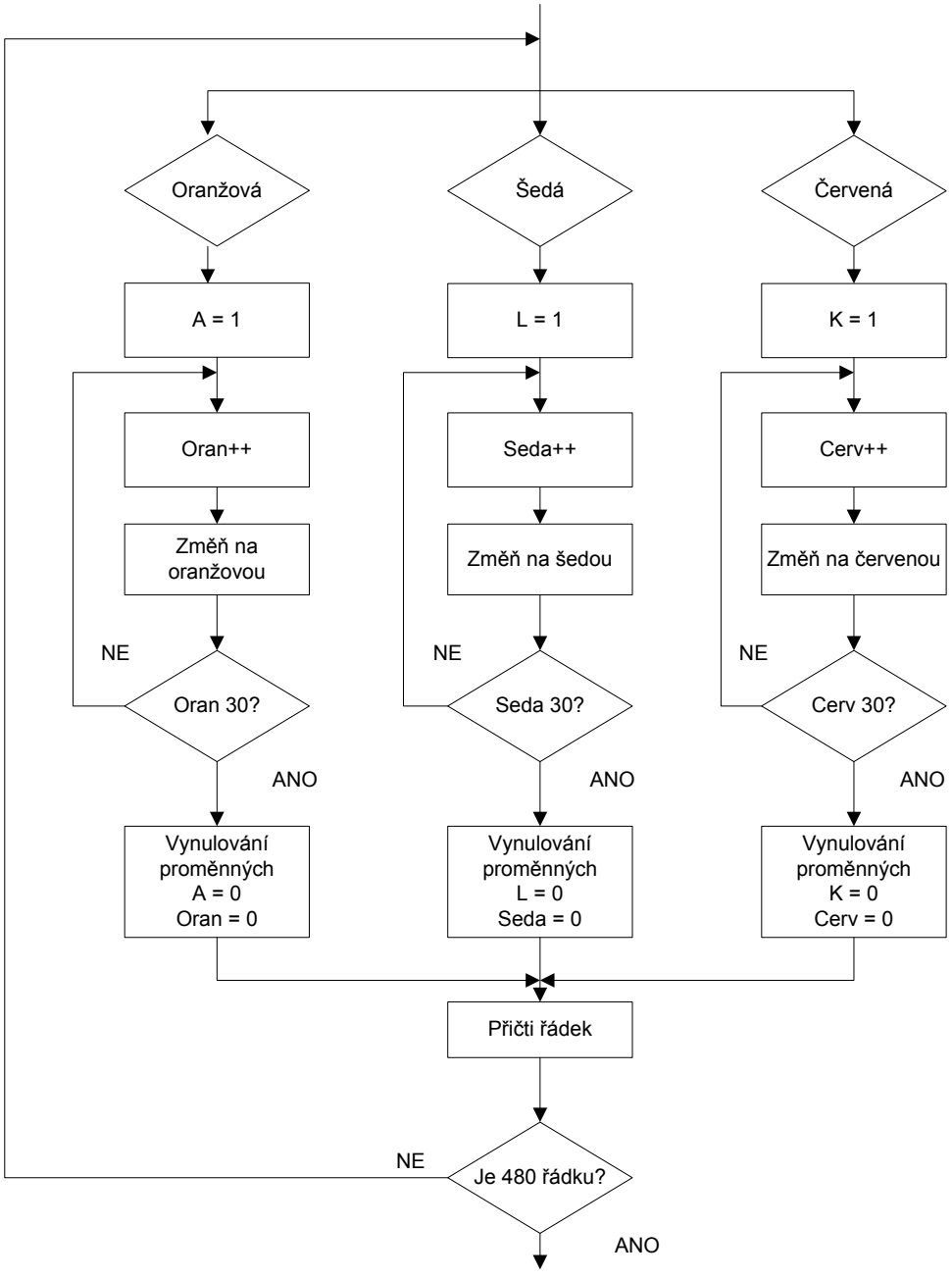
```
//oci
if ((c.R <= 100 && c.G <= 85 && c.B <= 70) && (c.R >= 80 && c.G >= 75
&& c.B >= 60))
```

Podmínka, která konvertuje barvy blízké lidským rtům do červené.

```
//rty
if ((c.R <= 125 && c.G <= 80 && c.B <= 70) && (c.R >= 110 && c.G >= 65
&& c.B >= 55))
```

6.5.4. **Rozmazání obrazu UML diagram**

Jelikož byla detekce očí a rtů velice náročná na vyhledání, zvláště detekce vzdálenějšího objektu, bylo nutné nějakým způsobem tyto objekty zvýraznit. Proto byl vyvinut algoritmus, který všechny tyto objevené a přepsané pixely do nově přebarvených barev, rozmazal o 30 pixelů doprava. Tím je dosaženo, že obličej získá širší obraz, ve kterém už je možná detekce a vyhledání středu.



Obrázek 35. UML rozmazání obrazu

Jsou rozmazány všechny barvy a tím se obličej celý posune o 30 pixelů. V případě objevení hledané barvy, se změní proměnná z hodnoty 0 na hodnotu 1. Tím je řečené, že následujících 30 pixelů bude překresleno, v tu stejnou barvu, jako nalezený pixel. Po překreslení všech požadovaných pixelů, se proměnná změní zpět na hodnotu 0, pokud nebyl objeven opět pixel se stejnou barvou. V takovém to případě se situace opakuje. Tato akce se nejčastěji objevuje u oranžové barvy, která signalizuje barvu lidské tváře, které je na obličeji nejvíc.

V UML diagramu je vidět, jakým způsobem algoritmus pracuje. Je to zde prezentováno jako jeden řádek. Proto není vidět připočítávání druhého vnořeného *foru*, jako přičtení následného pixelu. Je to z důvodu přehlednosti diagramu.

6.5.5. **Rozmazání obrazu kód v jazyce C**

Základem je posunutí hledaných barev o 30 pixelů doprava

Podmínka, když se objeví bílá barva, tak se proměnná *k,* změní z hodnoty 0 na 1. Pokud je 1, tak se připočítá proměnná *bila* a zároveň se každý další pixel mění v bílou, dokud se nenapočítá 30. Pak se *bila* a *k* vynuluje a hledá se další barevný pixel.

Pro příklad je zde uvedená pouze bílá barva, avšak další dvě barvy jsou, až na změnu hledané podmínky a proměnné, principiálně stejné.

```
//bila
if (c.R == 255 && c.G == 255 && c.B == 0)
                {
                    k = 1;
                }
```

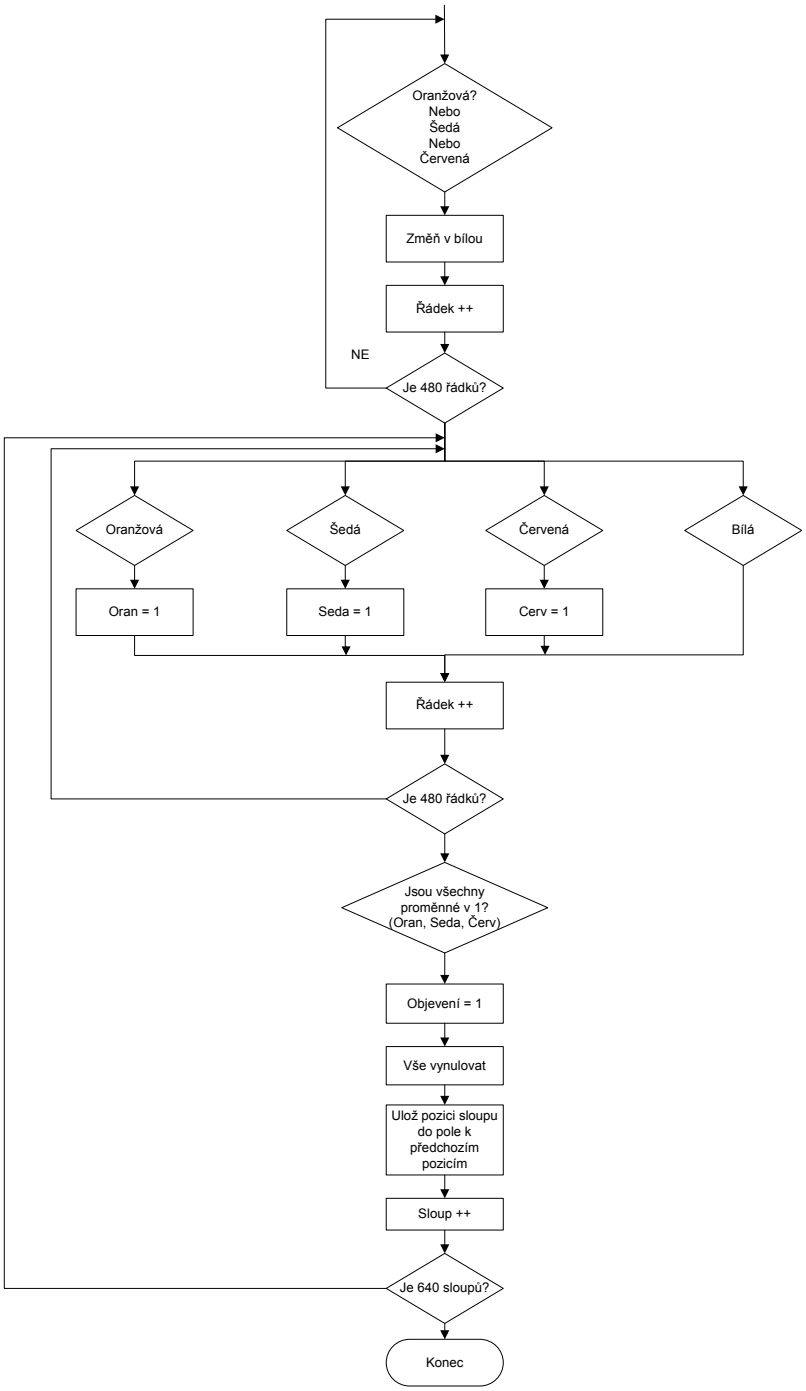
Když se objeví hledaná barva, proměnná se změní a následujících podmínka tuto změnu hledá. Když je podmínka splněna, začne se připočítávat proměnná sledující, aby bylo přepsáno jen 30 následujících pixelů.

```
if (k == 1)
                {
                    bila++;
                    c = Color.FromArgb(255, 255, 0);
                    sourceimage.SetPixel(m, n, c);
                }
```

Po dosažení hodnoty proměnné 30 se přestane překreslování pixelů a veškeré proměnné potřebné pro tuto akci se vynulují.

```
if (bila == 30)
                {
                    k = 0;
                    bila = 0;
                }
```

6.5.6. **Zvýraznění obličeje a objevení střed lidského obličeje UML diagram**



Obrázek 36. UML Zvýraznění a objevení středu obličeje

Když jsou veškeré barvy objeveny a rozmazány, je zbytečné a hlavně nepřehledné a matoucí mít v obraze i jiné barvy, proto se ostatní barvy změní v bílou. V podstatě se smažou.

Následující algoritmus neprohledává obraz po řádcích nýbrž po sloupcích. Tím je docíleno scan efektu.

Zároveň se z úspory výpočetního času, vyhledá střed obličeje. Když je detekována jedna z hledaných barev, změní se proměnná té barvy na hodnotu 1, tím je docíleno, že v jedné smyčce, byla tato barva detekována. Takto jsou hledány všechny barvy. Zároveň, pokud je barva jiná než tyto tři barvy, je pixel změněn na bílý. Když jsou nalezeny v jedné smyčce všechny při barvy, je pravděpodobné, že byla nalezena část obličeje. Změní se hodnota proměnné objevení na hodnotu 1 a pozice je uložena do pole. Proměnné potřebné pro výpočet se vynulují a prohledává se následující sloup. Opět je UML diagram vytvořen jako jeden sloup, proto není nikde připočítávána následující řádek.

6.5.7. Zvýraznění obličeje a objevení střed lidského obličeje kód v jazyce C

Následující kód převádí obraz do bílé. Je to z důvodu usnadnění detekce. Pokud je splněna podmínka, že se nalezne jedna z barev, do které byl obraz převeden, tj oranžová, červená nebo žlutá, tak se nic neprovede, jinak se vše změní v bílou.

```
if ((c.R == 255 && c.G == 0 && c.B == 0) || (c.R == 255 && c.G == 255
&& c.B == 0) || (c.R == 255 && c.G == 190 && c.B == 0))
      {

      }
else
      {
         c = Color.FromArgb(255, 255, 255);
         sourceimage.SetPixel(m, n, c);
      }
```

Následující kód vytváří v podstatě jakýsi druh scaneru. Každý sloup obrázku se provede zvlášť. Při objevení všech tří barev se do pole *poleXove* uloží pozice. Kód je tvořen převážně podmínkami s přesným nastavením RGB hodnot. Poslední podmínka zjišťuje, zda byly ve sloupci objeveny všechny barvy.

Až se narazí na konec sloupce, přičte se proměnná pro posunutí scanovaného sloupce a vynulují se proměnné, aby bylo možné počítat od začátku.

Když je objevení negativní, připočítá se proměnná jedna, aby bylo možné nalézt střed obličeje. K tomu je nutné znát i prázdné pixely.

Když je detekce na konci vynuluje se proměnná pro sloupy a zavolá se funkce, která určí, kde je střed obličeje. A všechny proměnné se vynulují pro další použití.

Funkce pro výpočet středu obličeje pracuje podobně, jako detekce středu. *Xová* souřadnice se spočítá tak, že

Princip vyhledání středu obličeje je stejný jako v kapitole 5.2. Vypočítání středu vybraného objektu.

Načtou se hodnoty kolik, sloupů bylo před nalezením prvního sloupce, který splňoval podmínku nalezení všech tří barev. K ní se přičte polovina pixelů, ve kterých byl obličej detekován. Takto vznikne *Xová* souřadnice. Dále se prohledá pole, ve kterém se vypočítá souřadnice *Y*. A tyto souřadnice se odešlou po sběrnici k dalšímu zpracování a následnému otočení hlavy směrem k obličeji.

6.6. Načtení obrazu ze senzoru do pole

Načtení obrazu se prování pomocí série pulsů náběžných hran. Princip je popsaný v kapitole 2.12 Čtecí sekvence a synchronizace pro kamerový senzor MT9V131.

První *while* čeká, dokud puls z *PIX CLK* nespadne na log 0. Druhý *while* zase čeká na náběžnou hranu téhož pulsu. Tímto je docíleno, aby se nezačal načítat do pole snímek z poloviny, ale od začátku. Cyklem *Do-While* je ošetřeno, aby bylo pole naplněno jedním snímkem. Na začátku cyklu je *while,* který čeká na log 0 z *LINE_VALID* a na konci čeká na log 1.

```
void ReadRow(unsigned int r)
{
      int n = 0;
      int i = 0;

//cteci sekvence

      while(PTC4_GetDir()){};       //dokud PIX CLK == 1
      while(!PTC4_GetDir()){};      //dokud PIX CLK == 0
      do{
            while(!PTC5_GetDir()){};            //dokud LINE_VALID == 0

            FileName[n]=PortB_GetDir();
            n++;
            FileName[n]=PortB_GetDir();
            n++;
            FileName[n]=PortB_GetDir();
            n++;
            FileName[n]=PortB_GetDir();
            n++;
            ...

            while(PTC5_GetDir()){};             //dokud PIX CLK == 1

       }while(r != n);
}
```

Takto napsaný kód přijme všechna data, která jsou mu v tomto čase poslána. Pokud by se načítala pouze Y složka z obou barev U i V bylo by 640 hodnot.

Pro tuto aplikaci je však toto nemožné z důvodu rychlosti mikroprocesoru, proto je načítána pouze každá druhá Y-ová složka k jedné barvě. Je tedy načtena jen polovina čísel. To je 320 hodnot. V dalším cyklu jsou načítány složky druhé barvy a výsledný obraz poté složený v PC do jednoho pole.

7. Měření a verifikace

V této kapitole je otestována celá aplikace. Od univerzální deky, její rychlosti zpracovávající přijímaných dat, přes rychlost kamerového senzoru až po měření a testování sejmutého obrazu v PC.

7.1. Měření času jednotlivých příkazů

Jelikož data z kamery přicházejí co mikročipu přísně sekvenčně, je nutné pro jejich synchronizaci vymyslet, algoritmus který bude načítat ta správná data.

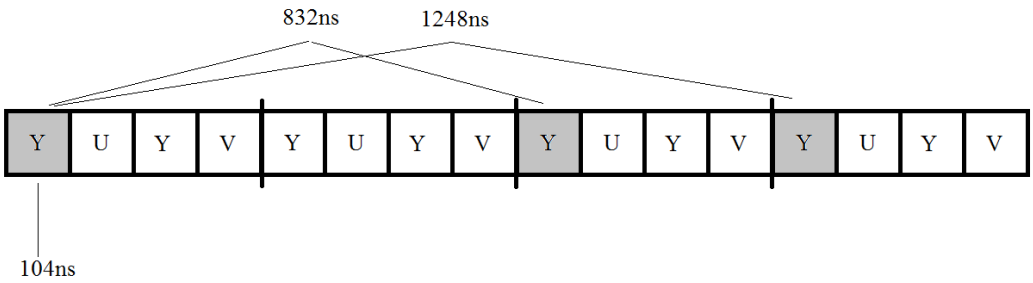
Název instrukce	**Čas změřený**	**Čistý čas**	**Změřená frekvence**	**Spočítaná frekvence**
LED	1,320µs	1,320µs	757,6 kHz	575,575kHz
Nop	1,480µs	160ns	675,7 kHz	675,675kHz
If	1,420 µs	100ns	704,2 kHz	704,225kHz
Hodnota+index	2,160 µs	840ns	463,0 kHz	462,962kHz
4x Nop	1,960 µs	640ns	510,2 kHz	510,204kHz
Odeslání po UART 320 hodnot	55,2ms	55,198ms	18,2Hz	18,11Hz
Do-While	1,960 µs	640ns	510kHz	510,204kHz

Název instrukce	**Čas změřený**	**Změřená frekvence**
Čas v řádku	0,134ms	7,440kHz
Čas mezi řádky	27,8µs	35,97kHz

Jeden řádek má 640pixelů. Každý pixel je tvořen dvěma čísly. To je buď Y+U nebo Y+V. Tzn., že na v době kdy je LINE_VALID v log 0 odešle kamerový senzor 1280 hodnot.

Pokud je doba, po kterou senzorů vysílá 0,134ms (7,4kHz) musí odeslat každou hodnotu v čase 104ns.

Pokud chci načítat pouze jasovou složku jedné barvy tj. pouze Y patřící k U a čas potřebný uložit jednu hodnotu trvá 840ns, můžu načítat každou 3tí hodnotu, která je vzdálená 1248ns.



Když mám 1248ns mezi požadovanými hodnotami a jednu hodnotu sejmu za 840ns musím čas potřebný k další hodnotě doplnit NOPy. Jelikož tato prázdná instrukce v jazyce C není je třeba jí do kódu přidat jako assemblerovský kód.

```
asm( NOP);
```

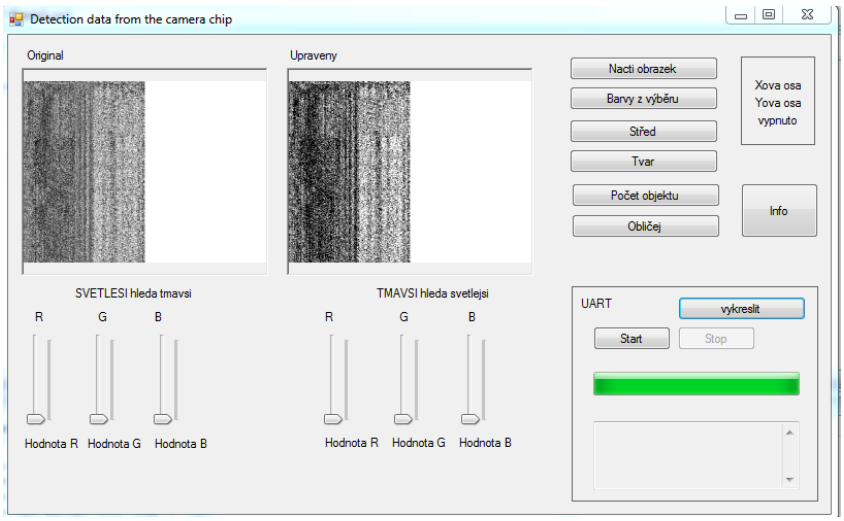
celkový čas odečtený od potřebného k sejmutí hodnoty
1248 – 840 = 768ns

Pokud prázdné místo vyplním čtyřmi NOPy a zbude čas 128ns.
768 – (4x160) = 128ns

7.2. Výsledný obraz z kamerového senzoru.

Kamera byla zaměřena na dvoubarevný papír, který měl jednu stranu černou a druhou bílou. Snímaný obraz byl ve vzdálenosti 7cm, tak jak je nastavena optika. Je snímán každý druhý jasový pixel patřící k U složce (sudá). Jsou vždy sejmuty dva řádky, odeslány do PC a následně sejmuty a odeslány liché jasové pixely.

Levý obrázek je originál přijatý obraz z kamery a pravý je vykreslený s důrazem na bílou a černou složku.



Je vidět, že do obrazu se míchají i jiné složky než Y. To je způsobeno špatnou sekvencí načítání anebo vnitřní funkcí mikroprocesoru, kdy může do kódu pro načítání zasáhnout vnější akce a tak načítání pozdržet nebo rozhodit.

7.3. Odesílání do PC po UART

Odesílám každý sudý pixel a pouze Y-ovou složku. To je 153 600 pixelů. Každý pixel má velikost 1Byte. Nastavená teoretická rychlost baud rate na UART je 115 200 bit/s

Výpočet:

115200 / 8 = 14,4kB/s

153 600 / 14400 = 10,66 s

Vzhledem ke špatné podpoře sériových linek v novějších operačních systémech Windows (Vista a vyšší) může být při použití převodníků USB/RS232 problematické až nemožné dosáhnout vyšší rychlost než 9600 bit/s. [15]

9600 / 8 = 1200B/s
153 600 / 1200 = 128s

Z -> kam	**Čas vypočítaný [s]**	**Čas změřený [s]**	**Reálná rychlost [Byte/s]**
MC -> UART	10:66	0:16:3	9423,3 (75386,4 bit/s)
MC-> PC	2:08	1:06:1	2323,75 (18590 bit/s)

Tedy 10,66sec se odešlou data z mikročipu na sériovou linku do PC a ten tyto data přijme až za minutu a 6,1 sec. Avšak v budoucnu se plánuje k tomuto mikročipu připojit RAM a doba potřebná k odeslání bude zcela irelevantní.

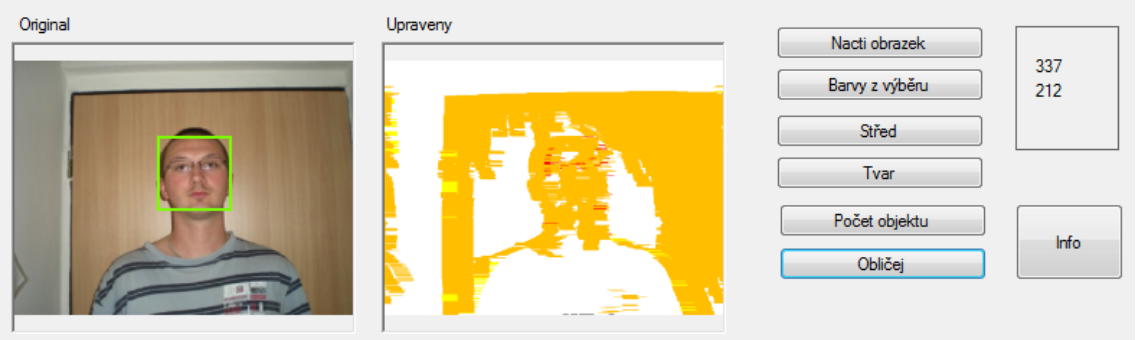
7.4. Měření přesnosti detekce obličeje

Jelikož je mikročip MCF51AC256 pomalý na snímání barevného snímku, je zde otestován algoritmus na detekci lidského obličeje pomocí jiné kamery, ale s podobnými parametry jako je rozlišení apod. Jedná se tedy o měření čistě algoritmu detekce lidských obličejů.

Snímek 1)

Osvětlení: žlutá žárovka

Pozadí: Dveře
Pohled: zepředu



Výsledek: X-337
Y -212
Výsledek je velmi přesný.

Snímek 2)
Jelikož s pozadím dveří byla detekce přesná, test bude spočívat detekci z úhlu.
Osvětlení: žlutá žárovka
Pozadí: Dveře
Pohled: Z úhlu

Výsledek: X-266
Y -191
Souřadnice ukazují přímo na střed hlavy, i když je vidět, že vlasy byly označeny za barvu rtů a proto byla hlava objevena v této pozici. Kdyby byla barva vlasů jiná, pravděpodobně by byl zelený čtverec blíže k tváři.

Snímek 3)
Osvětlení: Šero
Pozadí: Velmi barevné
Pohled: zepředu

Výsledek: X-265
Y -58

Obě souřadnice se mílí. Zkreslení je způsobeno pozadím, ve kterém se nachází mnoho barev, které se objevují i na lidském obličeji.

Snímek 4)

Osvětlení: bílá zářivka

Pozadí: šedé

Pohled: zepředu

Výsledek: X-339
Y -246

Byl objeven obličej. Střed hlavy však ne. Algoritmus detekce obličeje pracuje tak, že objeví jeden obličej a ten co je nejvíce vlevo nahoře. Nejblíže k souřadnicím 0,0.

8. Závěr

Cílem bylo vytvořit zařízení, které by sledovalo okolní svět a reagovalo na podněty z vnější, detekovalo a rozpoznávalo barvy, tvary a lidské obličeje.

Byly vytvořeny algoritmy hledající barvy. Tento algoritmus byl rozšířen o možnost nastavení rozmezí barev od světlejší po tmavší, což zpřístupnilo možnost detekce objektů nasvícených z jedné strany, kdy jedna strana byla jasnější a druhá tmavší. Návazně na tuto schopnost detekce byl vytvořen rozhodovací algoritmus detekující tvary. Hledaný tvar se nalezne podle přechodu mezi barvami, celý obraz se převede na černobílý formát a změřením jednotlivých stran objektu určí, o jaký tvar se jedná. Dále byl vytvořen algoritmus, počítající kolik objektů se nachází před objektivem kamery.

Vrcholem detekce je objevení obličeje. Zkombinováním různých funkcí kódu byl vytvořen algoritmus, hledající barvy, vyskytující se na lidském obličeji a z pozice těchto barev, vypočítá, kde se obličej nachází. Přesnost této aplikace byla testována a bylo prokázáno, že pokud se za detekovaným obličejem nenachází barevné pozadí obsahující barvy tváře, je detekce velice přesná.

Dalším aspektem této práce bylo vytvořit hardware s kamerovým senzorem, jeho periferiemi a připojením k univerzální desce. Tato deska byla navrhnuta, vyrobena a následně osazena optikou. Jelikož jsou univerzální desky uloženy v robotické hlavě, a kamerové desky měli prezentovat oči robota, musely být vyrobeny co nejmenší, aby se s nimi dalo v hlavě pohybovat. Současně k nim byl vyroben kabel spojující obě části. Avšak prototyp byl vytvořen zrcadlově, a tedy nebylo možné desku se senzorem spojit v robotické hlavě.

Při načítání dat z kamery docházelo k mnoha specifickým situacím. Data z kamerového senzoru jsou odesílány přísně sekvenčně, proto byl vytvořen algoritmus, který sledoval náběžné a sestupné hrany signálu přicházejícího ze senzoru a na ně se optimalizoval. I když byla rychlost hodinového impulzu na senzor nastavena přesně na 12MHz, podle doporučeného nastavení od výrobce, nastala situace, kdy rychlost sejmutí jedné hodnoty trvá 840ns a hodnoty z čipu přicházejí v intervalu 84ns, což je desetkrát rychlejší. Dalším specifikem byla rychlost odesílání dat ke zpracování do PC. Obě tyto překážky byly vyřešeny tak, že je snímán každý druhý jasový pixel, data jsou odeslána do PC a v následujícím cyklu ostatní jasové pixely. Tímto způsobem je možné data sejmout a zobrazit. Při zobrazení je zřetelné, že se do jasových složek pletou i barevné a tím není obraz čistý. Jednou z možností je použít RAM paměť. Tato možnost je však do budoucna připravena jako jedno z řešení.

Po spojení všech vytvořených částí a oživení je jasné, že po z optimalizování rychlosti snímaných dat ze senzoru bude bez problému detekce tvarů a spočítání předmětů před kamerou. Detekce obličeje není zatím možná, protože je nemožné načítat s těmito poměry barevný obraz.

9. Literatura

[1] *Data sheet: Technical Data. APTINA. www.aptina.com [online]. 2006 [cit. 2012-07-12].* Dostupné z: http://www.aptina.com/assets/downloadDocument.do?id=831

[2] *Data sheet: Technical Data. FREESCALE SEMICONDUCTOR. www.cache.freescale.com [online]. 2008 [cit. 2012-12-14].* Dostupné z: http://cache.freescale.com/files/32bit/doc/data_sheet/MCF51AC256.pdf

[3] *COOL WEBSERVIS. Netcam.cz [online]. [cit. 2012-12-14].* Dostupné z: http://www.netcam.cz/encyklopedie-ip-zabezpeceni/rozliseni-videa.php

[4] *STASANET.cz: Bezpečnostní technologie [online]. [cit. 2012-15-12].* Dostupné z: http://www.stasanet.cz/Zaklady-zapojeni-IP-kamer-a-optiky/

[5] *ELNIKA PLUS. www.elnika.cz [online]. 2005 [cit. 2012-15-12].* Dostupné z: http://www.elnika.cz/elnika.php?p=cze/objektivy

[6] *KOUKAAM a.s. www.http://www.koukaam.se [online]. [cit. 2013-10-1].* Dostupné z: http://www.koukaam.se/koukaam/downloads/Koukaatko_c25.pdf

[7] *Tabulka pro orientační určení velikosti ohniska objektivu. In: Www.eshop.comintech.cz [online]. [cit. 2013-01-28].* Dostupné z: http://eshop.comintech.cz/db/wysiwyg/Image/Objektiv%20tabulka.jpg

[8] *DOBEŠ, Michal. Zpracování obrazu a algoritmy v C#. 1. vydání. Praha: BEN-technická literatura, 2008. ISBN 978-80-7300-233-6.* Dostupné z: http://shop.ben.cz/113160

[9] *BOVIK, Al. Handbook of Image & Video Processing. Canada: AcademicPress, 2000. ISBN 0-12-119790-5.*

[10] *MAREŠ, Amadeo. 1001 tipů a triků pro C#. první. Brno: ComputerPress, a. s., 2008. ISBN 978-80-251-2125-2.*

[11] *Periodická tabulka: Krystalografie. Www.prvky.com [online]. [cit. 2013-02-04].* Dostupné z: http://www.prvky.com/krystalografie.html

[12] *YUV to RGB Conversion. Http://www.fourcc.org/ [online]. [cit. 2013-06-21].* Dostupné z: http://www.fourcc.org/fccyvrgb.php

[13] *CAN Bus. Www.canbuskit.com [online]. [cit. 2013-06-30].* Dostupné z: http://canbuskit.com/what.php

[14] *Freescale semiconducctor. www.freescale.com [online]. [cit. 2013-06-25].* Dostupné z: http://www.freescale.com/webapp/sps/site/homepage.jsp?code=TOWER_HOME

[15] *Microsoft support www.support.microsoft.com [online]. [cit. 2013-07-15].* Dostupné z: http://support.microsoft.com/kb/119579/cs

Seznam příloh

I. Schémata

Schéma zapojení ke kamerovým senzorům

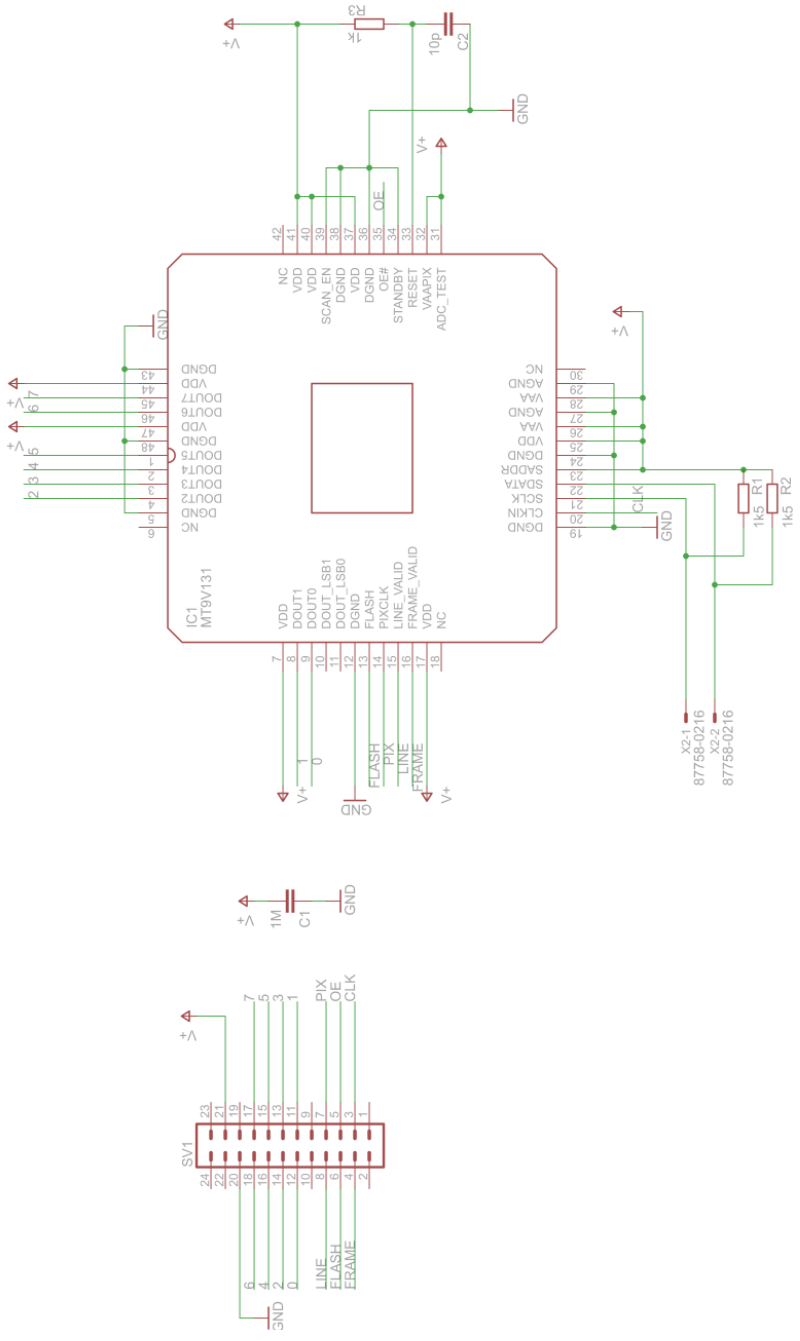
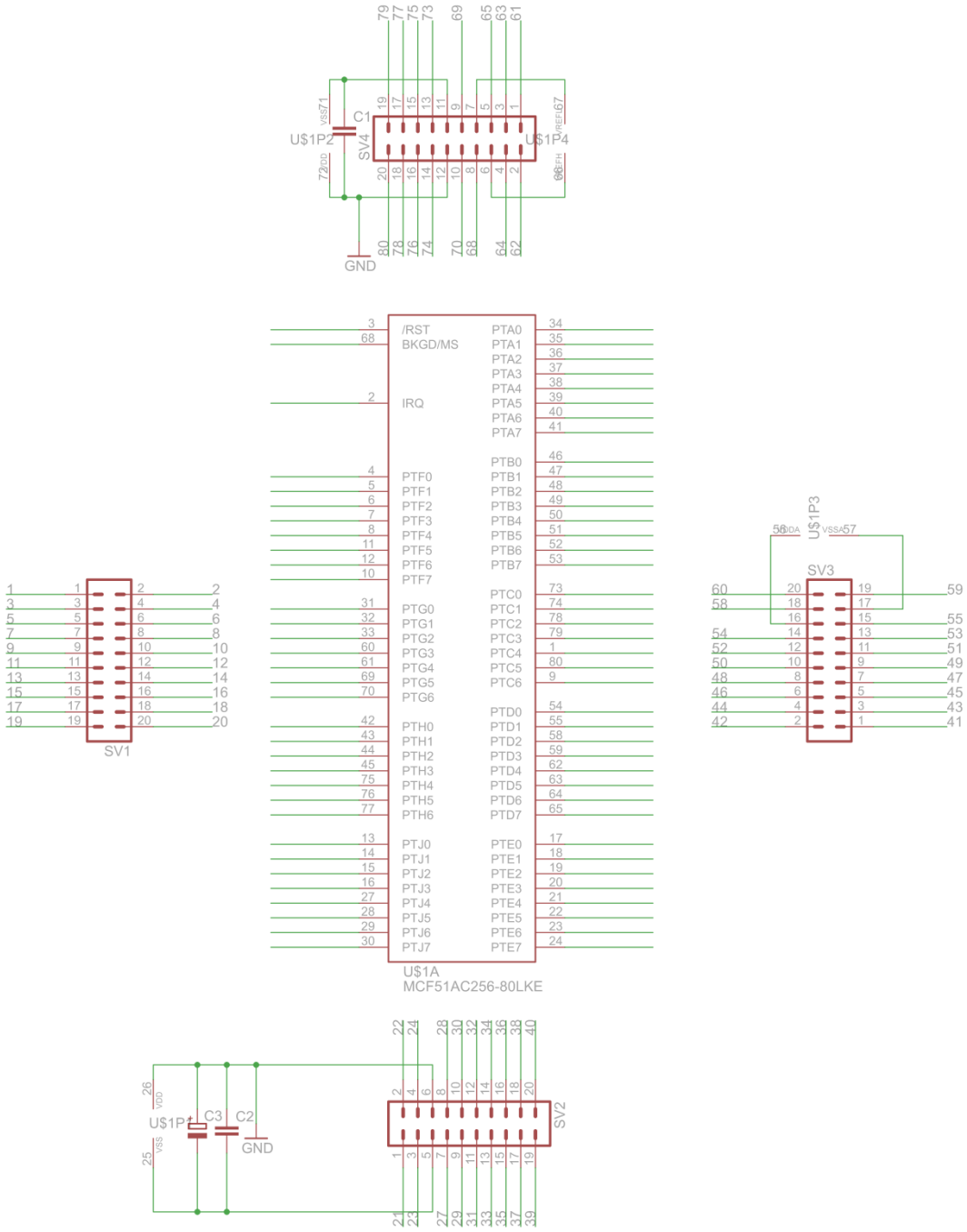
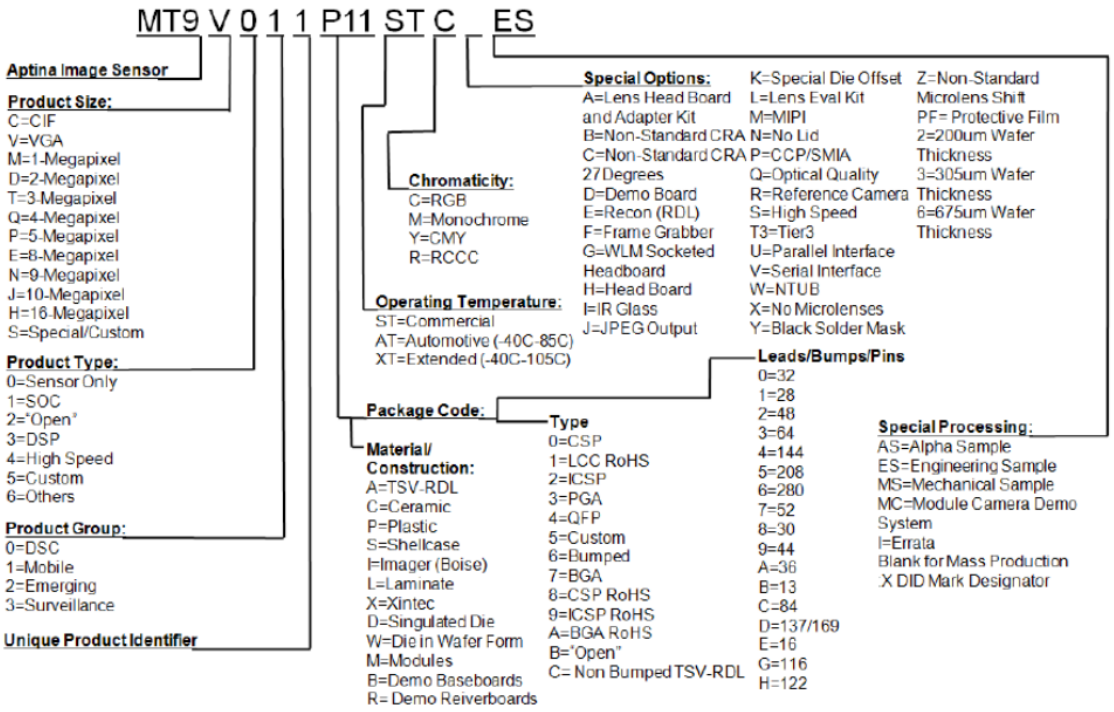


Schéma zapojení mikroprocesoru pro vývojový kit



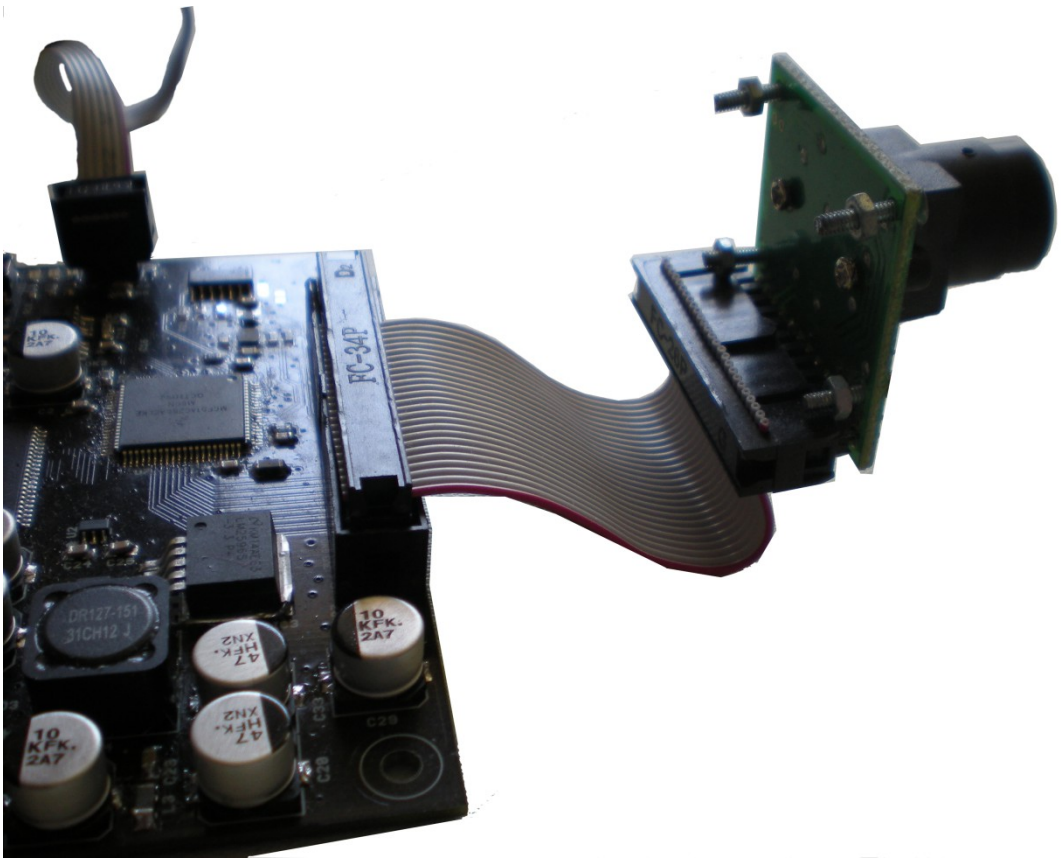
II. Označení kamerového senzoru



III. Fotodokumentace

Kamerový senzor na desce s optikou

Hlava, ve které jsou uloženy kamerové senzory



Upevnění kamerového senzoru na univerzální desce

IV. Obsah CD-ROM

- Diplomová práce
- Firmware pro kamerový senzor
- Kód ve Visual Studiu s GUI
- Kód ve Visual Studiu v GUI pro detekci lidského obličeje
- Datasheety k jednotlivým komponentám
- Video s detekcí barev, tvarů a počtu objektů
- Video s detekcí lidského obličeje
- Schéma a deska plošných spojů v Eagle 6.3.0
- Driver k PE micro Multilink Universal